Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Diseño e implementación de una cámara trampa de bajo coste

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: Aarón Vinent Pons

DIRECTOR: Eduard Garcia Villegas

FECHA: 14 de julio de 2017

Título: Diseño e implementación de una cámara trampa de bajo coste

Autor: Aarón Vinent Pons

Director: Eduard Garcia Villegas

Fecha: 14 de julio de 2017

Resumen

El objetivo de este proyecto es el diseño y desarrollo de una cámara on-ride, tipo cámara trampa de bajo coste, para su instalación en alguna atracción. La función de este tipo de cámaras consiste en tomar una fotografía automática en un punto de la atracción, normalmente en el momento de mayor adrenalina. Al salir de la atracción, el usuario puede visualizar o comprar la fotografía en un quiosco del parque. Su funcionamiento es sencillo, consiste en equipar a la atracción con un sensor que este enlazado y perfectamente sincronizado a la cámara para que, cuando se detecte movimiento, la cámara haga una fotografía de forma automática que posteriormente enviará al quiosco para poder descargarla.

Para poder realizar este dispositivo vamos a aprovechar las características, funcionalidades y bajo coste que presentan los ordenadores monoplaca como la Raspberry Pi, que a la que se instalará un sistema operativo Raspbian, el cual tiene una gran cantidad de librerías preinstaladas para proyectos de captura de imagen y para el manejo de sensores. El lenguaje de programación que utilizaremos para las aplicaciones será Python, que es un lenguaje fácil de aprender, muy popular y potente. Utilizaremos un editor nativo de Python como es IDLE.

Durante el proyecto se han implementado y evaluado tres sistemas de detección de movimiento diferentes. El primero, con un sensor infrarrojo pasivo (HC-SR501), el segundo con un sensor de ultrasonidos (HC-SR04) y, finalmente, un tercero que se realizará mediante software (procesado de imagen).

Para la captura de las fotografías se ha decidido utilizar el modulo cámara que dispone Raspberry Pi para la captura de imágenes. Este módulo toma imágenes de alta calidad de 8 megapíxeles.

Una vez probadas las diferentes soluciones propuestas, se escogerá un método seguro para la transmisión de las fotografías a un servidor externo. Como se realizaría en una cámara on-ride cuando transmite las imágenes al quiosco para la descarga y posterior venta.

Title: Design and development of a cheap trail camera

Author: Aaron Vinent Pons

Director: Eduard Garcia Villegas

Date: July, 14th 2017

Overview

The objective of this project is the design and development of a low-cost on-ride camera, a type of camera trap (or trail camera), for its installation in some attraction. The function of this type of cameras is to take pictures automtically at a certain point of the attraction, usually at the time of most adrenaline. When the user leaves the attraction, he or she can view or buy the photograph at a kiosk. Its operation is simple; the attraction is equipped with a sensor that is linked and perfectly synchronized to a camera so that a picture is automatically triggered when motion is detected to later send it to the kiosk to be able to download it.

In order to make this device, we will take advantage of the features, functionalities and low cost of the single-board computers such as Raspberry Pi, which will be installed the operating system Raspbian. It has a large number of libraries pre-installed for projects of motion detection and sensors. The programming language we will use for applications will be Python, which is an easy-to-learn, very popular and powerful language. We will use a native Python editor such as IDLE.

During the project, three different motion detection systems have been implemented and evaluated. The first with a passive infrared sensor (HC-SR501), an ultrasonic sensor (HC-SR04) and finally a third one to be realized by software (image processing).

For the capture of the photographs it has been decided to use the camera module that has Raspberry Pi for the capture of images. This module takes high quality images of 8 megapixels.

Once tested the different solutions proposed, we will be choosing as transmit the photographs to an external server securely. As would be done in on-ride camera, when it transmits the images to the kiosk.

INDICE

INTRODUCCIÓN

El proyecto nace de la idea de realizar una cámara trampa de bajo coste en comparación a las que existen actualmente en el mercado. Una cámara con unas características similares a la que realizaremos en nuestro proyecto tiene un precio aproximado de 290€ [1]. Concretamente nosotros vamos a tratar las cámaras on-ride, que son un tipo de cámaras trampa que se utilizan en los parques de atracciones, para realizar las fotografías de los usuarios. Estas cámaras tienden a instalarse en las atracciones más espectaculares y en la zona de máxima adrenalina. Requieren de una instalación fija y un estudio previo de la zona. Por este motivo, suelen realizarse proyectos especiales para su instalación y presentan diferentes precios según la complejidad y características. En el siguiente enlace [2] se puede observar una oferta pública para la instalación de dos sistemas on-ride completos en un parque de atracciones. En este proyecto, se pretende encontrar una solución equivalente de bajo coste y que permita la movilidad de poder instalarse y desinstalarse fácilmente para mejorar los servicios que se ofrecen, por ejemplo, en atracciones itinerantes.

Para la realización del proyecto, podemos sacar el máximo provecho de la versatilidad de una popular plataforma como la Raspberry Pi. Con una comunidad de desarrolladores muy activa, que hace muy fácil encontrar documentación para su uso en múltiples aplicaciones que requieren el uso de cámaras, sensores, LEDs, etc.

Durante el proyecto se han implementado tres tipos de soluciones para llevar a cabo la cámara on-ride. En primer lugar, se ha realizado el sistema con un método de detección mediante radiación infrarroja, utilizando para ello un sensor infrarrojo pasivo. En segundo lugar, se ha realizado un escenario en el cual la detección de movimiento es mediante procesado de imagen. Analizando los fotogramas para comprobar si se produce variación en la imagen. Finalmente, se ha realizado un escenario en el cual la detección de movimiento es mediante un sensor de ultrasonidos.

El plan de trabajo trazado para el desarrollo del proyecto, se desgloso en diferentes tareas que han estado vinculadas entre ellas para llegar al objetivo final del proyecto. Las tareas han sido las siguientes:

- Configuración hardware: Configuración de los dispositivos y obtención de un correcto funcionamiento de los sensores y la Raspberry.
- Configuración software: Instalación, configuración de la Raspberry y creación de las aplicaciones Python.
- Implementación: Diseño, desarrollo y resolución de los problemas de los escenarios.
- Documentación: Recopilación de toda la información y redacción de la memoria.

La presente memoria se ha dividido en seis capítulos. En el primer capítulo se hace una introducción sobre las cámaras trampa, donde también se explica su funcionamiento y aplicaciones.

El segundo capítulo contiene el entorno de desarrollo que se va a utilizar, este estará basado en la utilización de una Raspberry Pi y los componentes que utilizaremos para la realización del proyecto, además se realiza la descripción del software utilizado.

En el tercer capítulo se realizan las configuraciones comunes que tendrán los diferentes escenarios.

En el cuarto, se implementará el desarrollo de la solución que estará dividida en tres tipos de detección de movimiento. Una solución será implementarla con un sensor infrarrojo, otra solución con procesado de la señal y finalmente con un sensor ultrasónico.

En los últimos dos capítulos se evaluarán los escenarios desarrollados y se valoraran las conclusiones.

CAPÍTULO 1. ESPECIFICACIONES

En este capítulo, se pretende hacer una breve introducción al concepto de cámaras trampa y su funcionamiento. Así, como las aplicaciones que tienen en el mercado.

1.1. Cámaras trampa

Una cámara trampa se entiende como un dispositivo automático para capturar imágenes fotográficas, normalmente de animales en estado salvaje. Estos dispositivos integran una cámara fotográfica, sensores de movimiento y otros aparatos electrónicos que van encaminados a poder fotografiar o filmar en vídeo a personas o animales cuando aparecen delante de la cámara y hacen saltar el sensor.

Este tipo de dispositivos se instalan en un sitio difícilmente observado por la persona o animal que esperamos encontrar, esta técnica es conocida como fototrampeo, técnica muy empleada en la monitorización de poblaciones de animales.

Según el modelo, la cámara fotográfica se puede disparar al pisar una plataforma oculta en el suelo o mediante sensores de movimiento, esta última es la que nosotros utilizaremos.

Cuando el sensor de movimiento detecta la presencia de algún animal o persona, realiza una foto automáticamente. Algunos de estos dispositivos permiten tanto tomar fotografías, como capturar instantes de vídeo.

Su pequeño tamaño hace posible que pasen desapercibidas y que puedan estar vigilando durante las 24h del día sin que nada ni nadie las pueda detectar. Estos dispositivos consumen muy poca energía en modo de espera, de manera que pueden funcionar hasta 12 meses con un juego de baterías del tipo AA. Adicionalmente, se les puede conectar una batería externa para que duren más sin necesidad de cambiar las pilas.

Normalmente estos dispositivos, suelen instalarse dentro de una carcasa metálica para mejorar la protección contra elementos externos.

1.1.1. Funcionamiento

Estas cámaras están diseñadas para ser colocadas en plena naturaleza y quedar a la espera de que algún ser vivo pase por delante, momento en el que la cámara se activa y captura la escena.

Lo primero será empezar por su instalación. El primer paso a realizar es distribuir la cámara o cámaras y hacer un estudio de superficie. Para colocar la cámara hay que tener en cuenta la geografía del lugar. La cámara se dispondrá en un lugar donde no pueda ser arrastrada, golpeada o incluso robada. La posición del sol también será un punto a tener en cuenta ya que una exposición directa a la cámara podría producir destellos en las fotografías y quedar inutilizadas. Otros aspectos a tener en cuenta son la altura y el ángulo de la cámara, para poder obtener la zona de detección óptima. El sensor debe estar colocado estratégicamente sobre la zona de acción de la cámara para poder maximizar las posibilidades de realizar la foto deseada, siendo posible en algunos modelos tener separado el sensor de la cámara. Finalmente, para evitar falsos disparos por interferencias debidas a la temperatura, no se deberá dirigir el sensor infrarrojos hacia ninguna fuente de calor cercana. Cuando se utilice otro tipo de sensor de movimiento, se deberá calibrar la sensibilidad de los mismos para evitar activaciones innecesarias.

Una vez instalada la cámara, tocará poner en marcha el dispositivo y esperar a que los animales o personas pasen por delante. El diagrama de su funcionamiento es el siguiente:

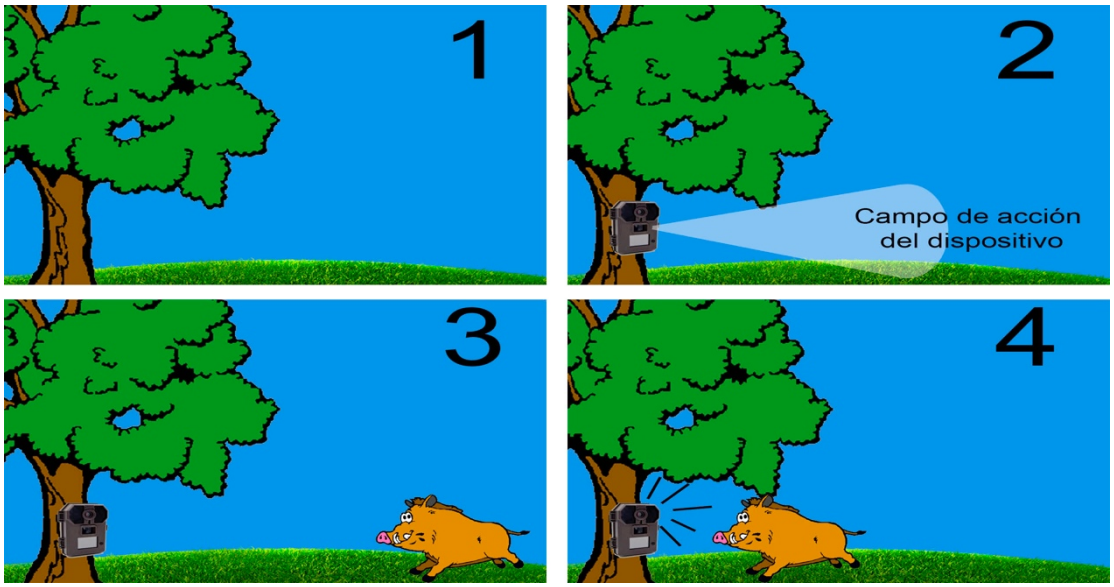


Fig. 1.1 Funcionamiento de una cámara trampa

1.1.2. Aplicaciones

Las cámaras trampa están teniendo una enorme aceptación en todo el mundo, tanto para aplicaciones de investigación biológica como para registrar animales cuya presencia es difícil de detectar. Este tipo de cámara es útil para estudiar el comportamiento animal, estimando patrones de actividad o realizando estudios de población.

Dos de las principales aplicaciones del fototrampeo son la cacería y la observación de la fauna silvestre.

Fig. 1.2 Imagen de un coyote realizada con una cámara trampa [3]

La gran ventaja de estas cámaras, comparándola con otros métodos de muestreo como la observación directa o captura, es que pueden registrar datos precisos sin que el animal tenga que ser capturado o que el observador esté presente.

Otra de las aplicaciones de gran interés es la de vigilancia de lugares alejados por motivos de seguridad, tales como fincas agrícolas, casas de campo, etc. Aunque también pueden utilizarse en sitios residenciales para controlar el acceso de las personas o para vigilar las propias viviendas.

Por último, comentar el uso de este tipo de cámaras para atracciones como las montañas rusas o parques acuáticos. En estos casos su funcionamiento es similar al comentado anteriormente, dispondremos de una cámara montada al lado de la pista de una montaña rusa (o atracción similar) que automáticamente toma fotografías de los pasajeros. Usualmente están montadas en la parte más intensa de la atracción, para realizar mejores capturas. Este tipo de dispositivo es conocido como cámaras on-ride.

Fig. 1.3 Imagen de una atracción realizada con una cámara on-ride [4]

CAPÍTULO 2. DESCRIPCIÓN DEL ENTORNO DE DESARROLLO

El objetivo de este capítulo es principalmente introducirnos en el mundo de la Raspberry Pi, de algunos periféricos disponibles para la plataforma y del software para controlar estos dispositivos. Primero se realiza una pequeña introducción de estos dispositivos y sus aplicaciones, para acabar detallando las especificaciones técnicas.

2.1. Raspberry Pi

La Raspberry Pi es un ordenador del tipo 'todo en uno' del tamaño de una tarjeta de crédito que se desarrolló en el Reino Unido por la *Fundación Raspberry Pi*, para promover la enseñanza básica de la informática y programación en las escuelas. En realidad, se trata de una pequeña placa base de 9x6 centímetros aproximadamente en el que hay un chip Broadcom con un procesador ARM, memoria RAM, una GPU, puertos USB y HDMI, entre otros, a un precio por debajo de los 40€. Esto último es uno de los principales motivos de su alta popularidad.

Su bajo coste, en parte, se debe a que utiliza software abierto, el sistema operativo oficial de la Raspberry se llama Raspbian, que es una versión adaptada de Debian, aunque también permite otro tipo de versiones Linux, así como versiones recientes de Windows.

Para que dicho ordenador pueda funcionar, necesitamos un medio de almacenamiento (una tarjeta de memoria de al menos 4GB), un cargador de corriente del tipo microUSB para poder alimentar nuestra Raspberry Pi.

En función del modelo que se escoja, dispondremos de diferente número de conexiones, pero todos los modelos tendrán al menos un Puerto USB 2.0, minijack, HDMI y un RCA. Respecto a las conexiones de red, la Raspberry Pi dispone de un Puerto Ethernet para poder conectar la Raspberry Pi directamente con un cable RJ-45, aunque hay modelos que ya tienen Wi-FI incorporado.

La Raspberry Pi dispone además de un sistema de conexiones denominado GPIO (General Purpose Input/Output) que se utilizan como entradas o salidas para usos múltiples, como la conexión de sensores.

Actualmente existen varios modelos desde que se lanzó en el año 2012, los modelos disponibles son los siguientes:

- **Raspberry Pi 1 Modelo A:** Este fue el primer modelo de Raspberry, se lanzó en Abril del 2012. No disponía de Puerto ethernet y necesitaba un adaptador Wi-Fi. Solamente disponía de un Puerto USB que se quedaba muy limitado para las conexiones. Es el modelo más básico de todos, su precio no superaba los 40€.

- **Raspberry Pi 1 Modelo B:** Lanzado también en el año 2012, es una variante mejorada del Modelo A, que trajo diversas mejoras consigo. Lo más significativo, fue que trajo un Puerto USB más y un conector Ethernet para RJ-45. Mantuvo su tamaño y precio.

- **Raspberry Pi 1 Modelo B+:** Lanzada poco después del Modelo B, este nuevo modelo mejorado, incluye 4 puertos USB y el almacenamiento pasó a ser mediante microSD.

- **Raspberry Pi 2 Modelo B:** Esta actualización en las Raspberry fue lanzada en el año 2014, la cual presenta mejoras sobre el modelo B+ y sigue manteniendo el número de pines y puertos USB, además suprimió el Puerto RCA. Este modelo mejora el procesador anterior, pasando a uno de cuatro núcleos a 900Mhz.

- **Raspberry Pi 3 Modelo B:** Esta nueva versión sale a la luz el año 2016. Mejora el procesador pasando a uno de 1200MHz. La gran novedad de esta nueva Raspberry es la inclusión de Wi-Fi y Bluetooth integrado (sin necesidad de adaptadores).

La tabla comparativa de los modelos disponibles en la actualidad es la siguiente:

Tabla 2.1. Tabla comparativa de modelos disponibles

Características / Modelo	Raspberry Pi 1 Modelo B+	Raspberry Pi 2 Modelo B	Raspberry Pi 3 Modelo B
Chip	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	ARM 1176JZF-S a 700 MHz	ARM Cortex A7 a 900 MHz quad-core	ARM Cortex A53 a 1,2 GHZ quad-core
RAM	256 MB	1 GB	1 GB
Almacenamiento	MicroSD	MicroSD	MicroSD
Puertos USB	4	4	4
Conectividad red	Ethernet (RJ-45)	Ethernet (RJ-45)	Ethernet (RJ-45), Wi-Fi 802.11n y Bluetooth 4.1
Consumo energético	600 mA	800 mA	800 mA
Precio	25€	35€	35€

2.1.1. Raspberry Pi 3 Modelo 3

Para nuestro proyecto utilizaremos la Raspberry Pi Modelo 3. Como se ha comentado en el apartado anterior, es el modelo más completo y avanzado que hay actualmente. Este último modelo salió a la venta, tras más de 8 millones de unidades vendidas de sus predecesoras y celebrando el cuarto aniversario desde su lanzamiento.

Las novedades más destacadas que trae este nuevo modelo es el nuevo procesador, un procesador de cuatro núcleos a 1200MHz de 64 bits y con un rendimiento muy superior al anterior (hasta un 50% más que el anterior). El otro gran cambio de este modelo es la conectividad inalámbrica, integrando conexión Wi-Fi y Bluetooth, sin necesidad de utilizar ningún periférico USB. A pesar de todos estos cambios, se sigue manteniendo el mismo tamaño y peso que las generaciones anteriores, únicamente cambiando algunos LEDs para la posición de la nueva antena.

Sobre el calentamiento de este nuevo modelo, desde *Fundación Raspberry* informan que, como los modelos anteriores, el dispositivo se calienta, pero que no es un factor peligroso para la maquina. Este calentamiento es normal y esperado, por lo que, generalmente, no necesitaremos un disipador de calor. Tras unos meses de uso hemos podido comprobar que este modelo se calienta y con un uso intensivo alcanza temperaturas muy altas, pudiendo llegar a los 100ºC, un valor muy alto que podría dañar o desgastar la propia placa. Esta temperatura se debe al cambio de procesador de las versiones anteriores.

Un procesador más potente necesita más energía y genera más calor. Al calentarse entra en modo protección de hardware que limita la velocidad, hasta que vuelve a tener una temperatura aceptable, produciendo una pérdida de potencia y rendimiento. Por estos motivos, en la práctica sería aconsejable utilizar un disipador de calor o utilizar una carcasa que ofrezca una adecuada ventilación.

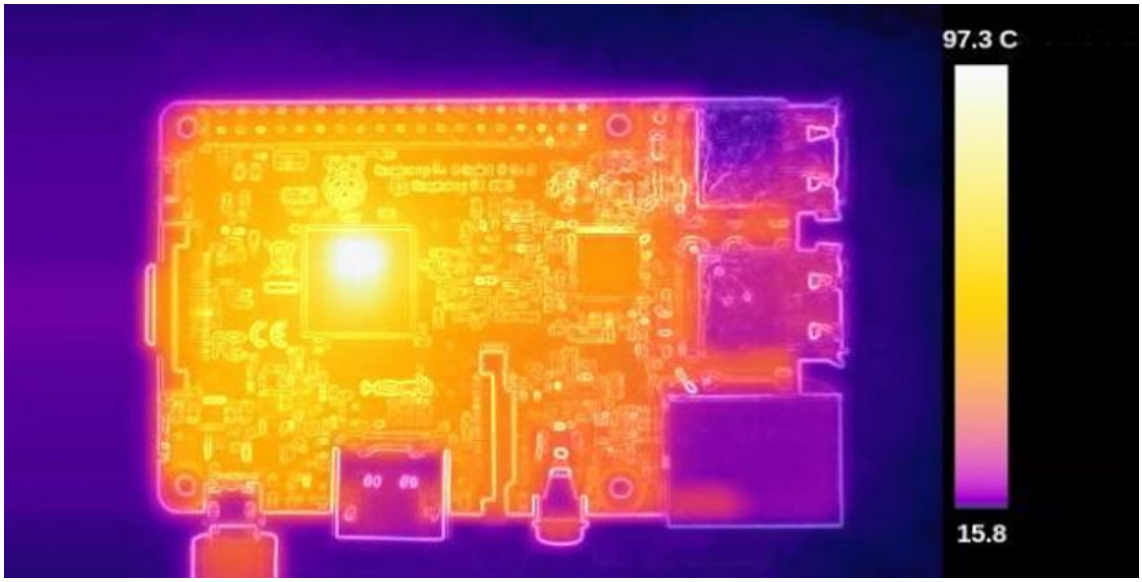


Fig. 2.1 Imagen termal de la Raspberry Pi [5]

2.2. Lista de Componentes

En el siguiente apartado se realizará una breve descripción de los componentes que utilizaremos a lo largo del proyecto para ejecutar los diferentes escenarios. En el siguiente enlace se pueden comprobar los periféricos que son compatibles con la Raspberry Pi [6].

2.2.1. Camera Module v2

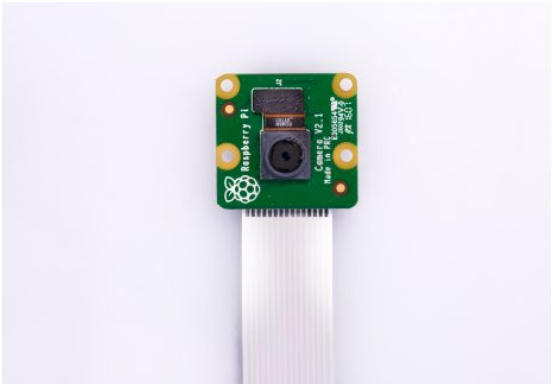


Fig. 2.2 Módulo Cámara v2

Esta versión 2 de la *Camera Module* reemplazó al módulo original en abril de 2016, este nuevo módulo tiene un sensor Sony IMX219 de 8Mpx, a diferencia de la v1 que tenía un sensor Omnivision OV5647 de 5 Mpx, que cuenta con enfoque fijo. Este módulo capta imágenes de 3280 x 2464 píxeles y vídeo de 1080 a 30 fotogramas por segundo, 720 a 60 fotogramas por segundo y 640x480 con hasta 90 fotogramas por segundo. Para conectar la cámara a la placa se realiza mediante un cable plano corto diseñado especialmente para el conector CSI (Camera Serial Interface) de que dispone la Raspberry. El tamaño de la cámara es de 25mm x 23 mm x 9 mm y tiene un peso único de 3 gramos, lo que hace un dispositivo ideal para aplicaciones portátiles. Es fácil de utilizar para principiantes, ya que hay una gran cantidad de ejemplos online para familiarizarse con el dispositivo y con sus librerías. Este módulo es muy popular para aplicaciones de seguridad, para cámaras trampa o para realizar efectos de imagen.

Las especificaciones del dispositivo son las siguientes:

Tabla 2.2. Especificaciones del módulo cámara

Especificaciones	
Sensor	CMOS Sony IMX219
Resolución	8 Mpx
Tasa de transferencia de la imagen	1080p: 30fps; 720p: 60fps; VGA90

Conexión	Cable plano de 15 pines
Dimensiones	25 x 23 x 9 mm
Peso	3 gramos
Compatibilidad Raspberry	Todos los modelos
Precio	30€

2.2.2. Sensor infrarrojo pasivo

Fig. 2.3 Sensor Infrarrojo Pasivo

Un sensor infrarrojo pasivo, a partir de ahora PIR sensor, es un sensor electrónico que mide la luz infrarroja (IR) radiada por los objetos (vivos o no) situados en su campo de visión. Estos sensores se utilizan principalmente como detectores de movimiento.

Estos dispositivos funcionan gracias a la radiación infrarroja que es imperceptible para el ojo humano, así pues, los objetos con una temperatura por encima del cero absoluto emiten un calor que puede ser detectado por estos dispositivos que han sido diseñados para tal fin.

Estos sensores únicamente detectan la energía emitida por otros objetos, en ningún momento genera o irradia energía para detectar los objetos, de ahí que se consideren dispositivos pasivos.

Los dispositivos PIR disponen de un sensor eléctrico capaz de captar dicha radiación y convertirla en señal eléctrica. Cada sensor está dividido en dos mitades de forma que detectan un cambio en la radiación infrarroja que reciben los dos lados. Si ambos campos reciben la misma cantidad de infrarrojos, la señal eléctrica será nula. En cambio, si los dos campos tienen una medición diferente, se genera una señal eléctrica.

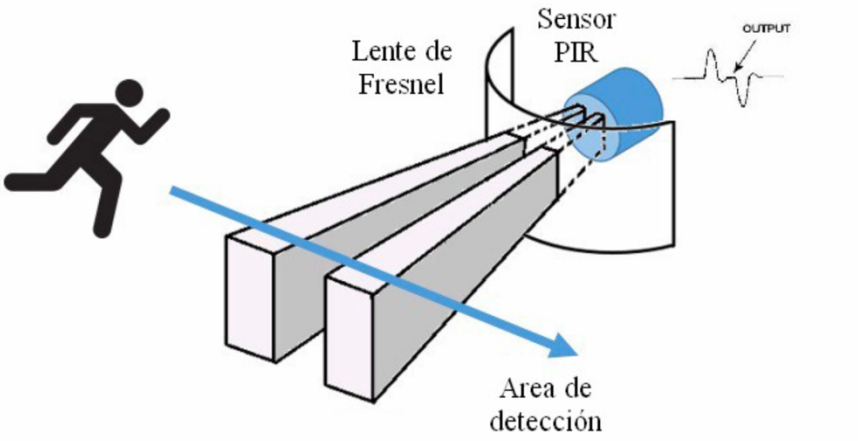


Fig. 2.4 Funcionamiento HC-SR501 [7]

Otro elemento que tienen estos dispositivos para que funcionen correctamente es la óptica del sensor, que consiste en una cúpula de plástico, formada por lentes de fresnel que dividen el espacio en zonas y enfocan la radiación a cada uno de los campos, para que así cada uno capte un promedio de la radiación del entorno. De esta manera, cuando un objeto entre dentro del rango del sensor, una de las zonas recibirá una cantidad distinta de radiación lo que activará la alarma.

Para nuestro proyecto utilizaremos el sensor HC-SR501, un sensor de bajo coste, donde la señal generada por el sensor ingresa al circuito integrado BISS0001, que contiene amplificadores operacionales e interfaces electrónicas.

Para poder ajustar los parámetros del sensor disponemos de 2 potenciómetros, que el usuario puede modificar para variar tanto la sensibilidad como la distancia de detección del sensor.

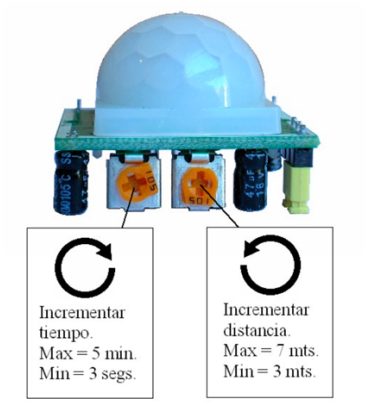


Fig. 2.5 Potenciómetros del sensor para ajustar parámetros [5]

Tal como se ha comentado en el párrafo anterior, los sensores son ajustables y normalmente funcionan con unos alcances de hasta 7 metros y una apertura de entre 90º y 110º.

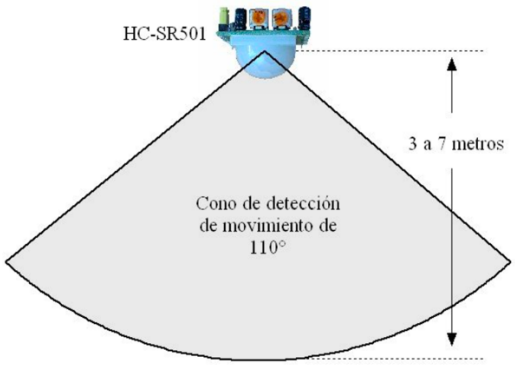


Fig. 2.6 Cono de detección del sensor HC-SR501 [5]

Los pines del sensor son los siguientes:

- VCC, es el terminal de alimentación que debemos conectar un voltaje.
- GND, es el terminal de tierra donde conectaremos a un voltaje cero para completar el circuito.
- OUT, es el pin por el cual el sensor nos contestará una vez detecte movimiento.

Las especificaciones del sensor son las siguientes:

Tabla 2.3. Especificaciones del sensor HC-SR501

Especificaciones	
PIR	LH1778
Controlador	BISS0001
Voltaje	5-12 v
Rango distancia	3-7 metros
Angulo de detección	110º
Vo	3,3 v
Dimensiones	3,2 x 2,4 x 1,8 cm
Precio	6 €

2.2.3. Sensor de Ultrasonidos

Fig. 2.7 Sensor de Ultrasonidos

Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. Son detectores de proximidad que trabajan libre de roces mecánicos y pueden detectar objetos desde pocos centímetros hasta varios metros, según el modelo. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna cuando un objeto pasa por delante del sensor y pueden medir la distancia al objeto contando el tiempo entre la emisión y la recepción de la señal, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

Existen dos tipos de configuraciones para realizar estos sensores: con sistema de emisor/receptor en el mismo transductor y con el sistema emisor/receptor separados, este último es el que utilizaremos para realizar el tercer escenario (sección 4.3). En esta configuración, las ondas se generan en una de las membranas, mientras que en la otra, escucha desde el mismo momento en el que se emiten las ondas. Suele ser apropiado para medir distancias cortas que la otra configuración no puede alcanzar.

Para medir la distancia se utiliza la técnica de tiempo de vuelo, que consiste en emitir un impulso y poner un temporizador en marcha, cuando se recibe el eco de los impulsos emitidos; el tiempo transcurrido es proporcional al doble de la distancia al objeto. Si no se recibe el eco, se considera que no hay obstáculo.

Para nuestro proyecto utilizaremos el sensor HC-SR04, este módulo tiene un rango de funcionamiento de entre 3cm y 3m y se alimenta con 5V. Tal y como hemos explicado anteriormente, su funcionamiento es el que se muestra en la Fig. 2.8:

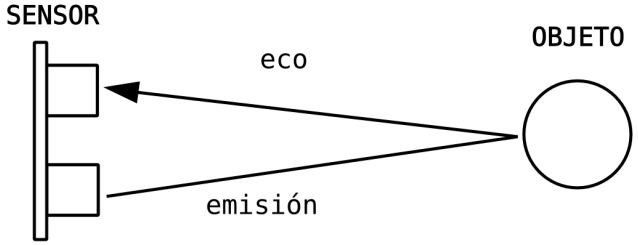
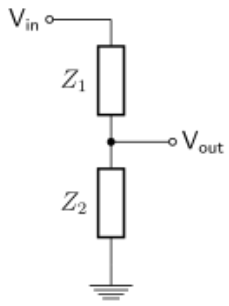


Fig. 2.8 Funcionamiento HC-SR04

Un aspecto importante a tener en cuenta al utilizar este modelo de sensor en una Raspberry Pi, es que funciona con 5V, tanto para su alimentación como para su salida hacia la Raspberry, la cual trabaja a 3,3V. Para solucionar el problema y no estropear nuestro dispositivo, tendremos que hacer un divisor de tensión entre la salida del sensor y la Raspberry, de esta manera podremos fijar una tensión intermedia entre el nivel de alimentación y el nivel de tierra. El esquema y la formula para calcular el voltaje requerido lo podemos ver en la siguiente imagen:



$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Fig. 2.9 Esquema y formula del divisor de tensión

El sensor tiene los siguientes pines:

- VCC, es el terminal de alimentación que deberemos conectar a un voltaje de 5V.
- Trig, es un pin de control por el cual la Raspberry enviara una señal al sensor, para que mida la distancia al objeto que tenga enfrente. Es un pin de entrada y con un voltaje de 3,3V funcionará correctamente.
- Echo, es el pin a través del cual el sensor contestara a la Raspberry cuando tenga respuesta, haciendo variar su salida entre 0 y 5V.
- Gnd, es el terminal al que deberemos conectar un voltaje de 0V para poder cerrar el circuito.

Las especificaciones del sensor son las siguientes:

Tabla 2.4. Especificaciones del sensor HC-SR04

Especificaciones	
Alimentación	5V
Frecuencia	40 KHz
Ángulo efectivo	15º
Distancia	3cm a 3m

Resolución	3mm
Dimensiones	45 x 20 x 15 mm
Precio	6 €

2.2.4. Otros

Aparte de los materiales comentados anteriormente, también deberemos tener en cuenta los siguientes accesorios a la hora de poner en marcha la Raspberry y posteriormente establecer nuestros escenarios. Lo que necesitaremos será lo siguiente:

- Tarjeta MicroSD: Las tarjetas deben tener al menos 8 GB de tamaño, las más antiguas y lentas no funcionan como deberían. Por lo tanto las tarjetas de clase 4 no son compatibles y por esta razón se deben utilizar como mínimo las de clase 10.

Fig. 2.10 MicroSD

- Fuente de alimentación: La fuente de alimentación recomendada para la Raspberry Pi se compone de un cable integrado y un conector microUSB con una alimentación de 5V y 2A.

Fig. 2.11 Fuente de alimentación

2.3. Software

La Raspberry Pi no es compatible con todos los sistemas operativos que utilizan los ordenadores, debido a que el tipo de arquitectura no es la misma que los ordenadores convencionales. A pesar de esto, los desarrolladores han creado sistemas operativos específicos para la Raspberry Pi.

Actualmente, hay una gran cantidad de sistemas operativos y gestores de contenidos que puedes instalar en la Raspberry, aunque la lista es muy amplia, se comentaran los más comunes.

- Raspbian: Es el sistema operativo que soporta la Raspberry Pi de manera oficial, está basado en Debian y es uno de los más populares.
- Arch Linux: Otra distribución Linux para la Raspberry, es un sistema ligero y con un diseño simple y minimalista ideal para esta Raspberry.
- Cent OS: Distribución para servidores que está disponible para Raspberry, es una variante de la familia Red Hat.
- Windows IoT: Versión especial de Windows para Raspberry Pi, no es una versión completa del sistema operativo.

2.3.1. Raspbian

Fig. 2.12 Logotipos de Raspberry y Raspbian

Raspbian es una distribución libre basada en Debian Jessie para la Raspberry Pi, dicho sistema fue lanzado a mediados de 2012 y está orientado a la enseñanza.

Dicha distribución contiene herramientas de desarrollo como IDLE para el lenguaje de programación Python o Scratch y diferentes ejemplos usando módulos.

Otra opción que existe es la 'raspi-config' que tiene la posibilidad de configurar el sistema operativo de manera sencilla para cualquier usuario, entre las que destacan la posibilidad de expandir la partición, configurar teclado o configurar entradas, etc.

Al tratarse de una distribución Linux tenemos una gran cantidad de software libre que puede ser aplicado en la propia Raspberry. Además, dispone de repositorios donde el usuario puede descargar una gran cantidad de programas para equipos de escritorio haciendo posible que la Raspberry sirva como un microcontrolador clásico o como un ordenador personal sencillo y de bajo coste.

Las características de este sistema operativo son:

Tabla 2.5. Especificaciones de Raspbian

Especificaciones	
Desarrollo	Software libre
Lanzamiento	Junio 2012 (última Junio 2017)
Núcleo	Linux
Interfaz gráfica	LXDE
Plataforma	ARM
Programas por defecto	LibreOffice, Navegador web, Herramientas programación

2.3.2. Python

Fig. 2.13 Logotipos de Python

Python es un lenguaje de programación popular, fácil de aprender y muy completo. Tiene una estructura de datos de un alto nivel y es efectivo para la programación orientada a objetos. Es un lenguaje ideal para realizar scripting y desarrollo rápido de aplicaciones diversas.

Además, es desarrollado bajo una licencia open-source, por lo que puede ser utilizado en cualquier sistema con total libertad. Esto es uno de los motivos por lo que *Fundación Raspberry Pi,* lo ha elegido como lenguaje de programación preferente para sus dispositivos, aunque también pueden ejecutar programas escritos en cualquier otro lenguaje.

Python es un lenguaje interpretado, que necesita un software que se encarga de convertir las sentencias y las ejecuta a tiempo real.

En Raspbian, el software que viene preinstalado para el desarrollo en Python es IDLE (Integrated DeveLopment Environment for Python) un entorno de desarrollo que permite editar y ejecutar las aplicaciones creadas. Este software permite lo siguiente:

- Editor de texto multiventana

- Destacado de la sintaxis de la programación
- Función de autocompletar
- Depurador integrado

2.3.3. Secure Shell (SSH)

SSH es un protocolo que ayuda a facilitar las comunicaciones entre dos sistemas, usando una arquitectura cliente/servidor y permitiendo una conexión a un host remoto. Este tipo de conexión encripta la sesión de manera que nadie puede conocer contraseñas no encriptadas ni interceptar los datos transmitidos. Es un protocolo diseñado para reemplazar métodos más antiguos y menos seguros como telnet, que no ofrecen ningún tipo de seguridad.

Este protocolo ofrece características muy buenas, que han hecho que se convierta en el método más utilizado por los usuarios para gestionar servidores Linux remotamente. Dichas características son:

- El uso de SSH encripta la sesión de registro impidiendo que nadie pueda conseguir las contraseñas encriptadas.
- Las claves únicamente son conocidas por quien emite la información y quien la recibe.
- Cualquier alteración en la clave modifica el mensaje original.
- El usuario tiene posibilidad de verificar que sigue conectado.
- Cuando se realiza la autentificación, se crea un canal seguro cifrado para el intercambio de información.
- Los datos se encriptan por medio de algoritmos de encriptación de 128 bits.

2.3.4. VNC

VNC es una programa de software libre basado en la estructura cliente/servidor que permite tomar el control de un servidor remotamente a través del cliente, también es denominado software de escritorio remoto. Esta aplicación no impone restricciones con el sistema operativo del servidor con respecto al del cliente, se puede compartir una pantalla de una máquina virtual con otro sistema operativo mientras admita VNC.

2.3.5. Secure File Transfer Protocol (SFTP)

SFTP es un protocolo a nivel de aplicación que permite la transferencia de archivos sobre un canal de datos fiable. Es un protocolo de transferencia que utiliza SSH para asegurar la transferencia de datos entre cliente y servidor y que nadie no autorizado tenga acceso a dichos datos.

CAPÍTULO 3. PREPARACIÓN DEL ENTORNO

En este apartado se va a realizar la preparación de los aspectos comunes que posteriormente se utilizaran los tres escenarios en el capítulo 4. Para dicha preparación se va a instalar el sistema operativo en la Raspberry Pi y se va a configurar para poder empezar con la implementación de los escenarios.

3.1. Instalación Raspbian

Lo primero que se ha de realizar para preparar el entorno en el que trabajaremos será instalar el sistema operativo en la Raspberry.

Para empezar, nos dirigiremos a la pagina oficial de Raspberry y entramos a la zona de descarga para descargarnos la última versión estable de Raspbian, que en el caso de este proyecto es del 11 de Enero de 2017. Una vez descargado deberemos preparar la tarjeta microSD para instalarle el sistema operativo con el que correrá nuestra Raspberry. Todos los comandos previos para preparar la tarjeta se han realizado desde una máquina con el sistema operativo Mac OS X, por lo que si se realiza con Windows o Linux el proceso puede ser distinto. En el caso de usar alguno de estos sistemas operativos, el lector puede consultar en [8] ó [9].

Se formateara la tarjeta al formato FAT32. Deberemos introducir la tarjeta en nuestro ordenador y buscar el identificativo que le asigna el ordenador a esta. El comando a realizar desde el terminal será el siguiente.

```
MacBook-Air:~ AnaGab$ diskutil list
/dev/disk0 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *121.3 GB   disk0
   1:                        EFI EFI                     209.7 MB   disk0s1
   2:          Apple_CoreStorage Macintosh HD            120.5 GB   disk0s2
   3:                 Apple_Boot Recovery HD             650.1 MB   disk0s3

/dev/disk1 (internal, virtual):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:                  Apple_HFS Macintosh HD           +120.1 GB   disk1
                                 Logical Volume on disk0s2
                                 D7D381A2-D1D2-4ADE-856C-B8ED10B5F1DF
                                 Unlocked Encrypted

/dev/disk2 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *15.9 GB    disk2
```

Como podemos observar, nuestra tarjeta microSD tiene el identificativo disk2, el cual deberemos desmontar para proceder a instalar el sistema operativo con el comando `diskutil unmountDisk /dev/disk2`. Tras desmontar la unidad de la microSD, deberemos insertar el sistema operativo que previamente hemos

descargado a la tarjeta. Se puede realizar a través de una aplicación o por el terminal que será el que utilizaremos. Los comandos son los siguientes:

```
MacBook-Air-de-Ana-2:~ Aaron$ cd Desktop/
MacBook-Air-de-Ana-2:Desktop Aaron$ sudo dd bs=1m if=2017-01-11-raspbian-
jessie.img of=/dev/rdisk2
4169+0 records in
4169+0 records out
4371513344 bytes transferred in 326.348701 secs (13395222 bytes/sec)
MacBook-Air-de-Ana-2:Desktop Aaron$
```

Una vez tenemos el sistema operativo en la tarjeta microSD, el siguiente paso será insertarla en la Raspberry Pi, conectar esta misma a un monitor con un cable HDMI y un ratón al puerto USB. Tras esto, deberemos enchufarla a la corriente para comprobar que ha sido correctamente instalado el sistema operativo en la microSD. El siguiente paso será conectar el dispositivo con un cable RJ-45 a un switch Ethernet con acceso a un servidor DHCP (e.g. router ADSL en red doméstica) para que se le asigne una IP dentro de nuestra red.

3.2 Configuración Raspberry

Una vez se ha iniciado la Raspberry, será momento de configurarla, por lo que nos dirigiremos al menú → Preferences → Raspberry Pi Configuration → Interfaces y habilitamos los puertos para la cámara, conexión ssh y para VNC tal como muestra la siguiente imagen.

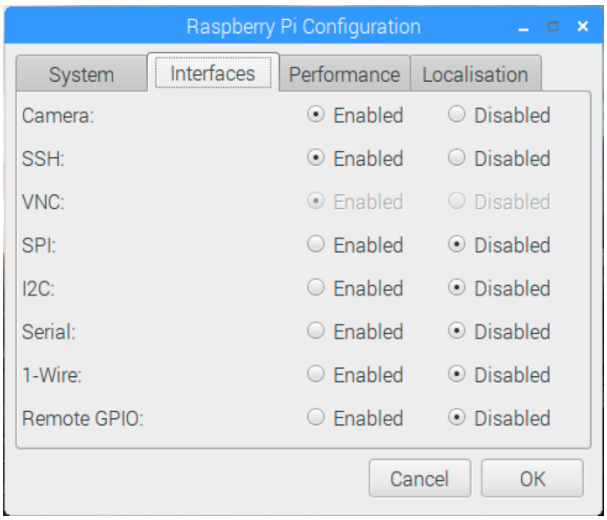


Fig. 3.1 Panel de configuración de la Raspberry Pi

Una vez hemos habilitado la conexión por ssh y por VNC ya podremos conectarnos en remoto con el ordenador sin depender de ningún periférico, mientras esté conectado a nuestra red. Antes de conectarnos mediante ssh deberemos conocer la IP que se le ha asignado a la Raspberry, por lo que abriremos el terminal y con el comando *ifconfig* podremos comprobar la IP que se ha asignado, en nuestro caso la 192.162.0.154.

Lo próximo será conectarse desde el ordenador en remoto a la Raspberry, para ello utilizaremos los siguientes comandos:

```
MacBook-Pro-de-Aaron:~ aaronvinentpons$ ping 192.168.0.154
PING 192.168.0.154 (192.168.0.154): 56 data bytes
64 bytes from 192.168.0.154: icmp_seq=0 ttl=64 time=6.927 ms
MacBook-Pro-de-Aaron:~ aaronvinentpons$ ssh pi@192.168.0.154
The authenticity of host '192.168.0.154 (192.168.0.154)' can't be established.
ECDSA key fingerprint is SHA256:hdDnMLWi/V8G+k7IzVtv/IyRTQECiV8MXpWStmLa3H0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.154' (ECDSA) to the list of known hosts.
pi@192.168.0.154's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun  7 22:51:08 2017

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to
set a new password.

pi@raspberrypi:~ $
```

De esta manera, primero se ha podido comprobar la conexión de la Raspberry con un ping y posteriormente se ha realizado la conexión mediante ssh. La contraseña que trae la Raspberry por defecto es ‘Raspberry’. Una vez dentro, es recomendable instalar todas las actualizaciones del sistema con los comandos:

```
pi@raspberrypi:~ $  sudo apt-get update
pi@raspberrypi:~ $  sudo apt-get upgrade
```

3.2.1 Instalación VNC

Para poder tener acceso remoto de manera gráfica mediante un escritorio remoto, necesitaremos hacer uso de la aplicación VNC, anteriormente le hemos permitido acceso mediante esta aplicación pero ahora deberemos instalar la aplicación, por lo que en el terminal utilizaremos el siguiente comando:

```
pi@raspberrypi:~ $  sudo apt-get install tightvncserver
```

Una vez instalado el servidor en la Raspberry, nos dirigimos a la página de VNC y nos descargamos la versión de nuestro sistema operativo [10]. Cuando ya la tengamos instalada, para poder acceder a la Raspberry deberemos entrar en el terminal de esta última y poner el siguiente comando:

```
pi@raspberrypi:~ $ tightvncserver

New 'X' desktop is raspberrypi:1
```

```
Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log
```

Con este comando la Raspberry ya está lista para conectarse mediante VNC, por lo que en la aplicación cliente deberemos configurarla con la IP de la Raspberry (192.168.0.154 en nuestro caso). El número '1' del último comando que aparece en la consola se refiere al display, que deberá ser el mismo que en el cliente.

3.2.2 Creación de programas con Python

Para poder correr nuestros programas en la Raspberry deberemos saber cómo hacerlo, para ello utilizaremos el IDLE de Python que viene preinstalado en la distribución Raspbian ejecutándose en nuestra Raspberry. Para ello nos dirigimos al menú → Programming → Python 3 (IDLE) como se muestra en la siguiente imagen.

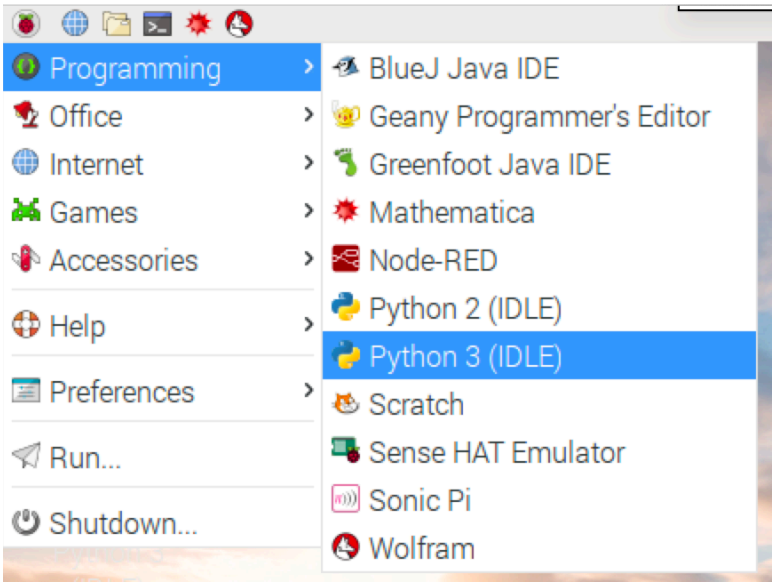


Fig. 3.2 Menú del sistema operativo Raspbian

A continuación, aparecerá la ventana principal de IDLE, esta nueva ventana aparece con la versión de Python instalada, en nuestro caso la versión 3.4.2 como muestra la imagen.

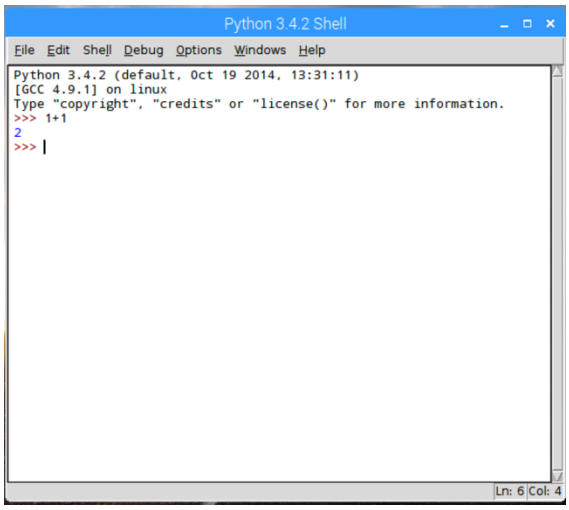


Fig. 3.3 Ventana IDLE de Python

En la ventana de IDLE se pueden escribir ordenes de Python tras los símbolos de petición o prompt en ingles (>>>). Al pulsar intro, el IDLE ejecutará la orden inmediatamente, si la orden se produce correctamente se mostrará en color azul y sin el prompt como podemos observar en la figura 3.3.

Si simplemente queremos ejecutar órdenes sencillas, la ventana del intérprete, es suficiente, pero si queremos realizar un programa más complejo será conveniente guardar el programa para así poder recuperarlo en cualquier momento. Para crear un archivo de programa, hay que ir a la ventana principal y abrir una nueva ventana mediante el menú File → New File, de esta manera nos aparecerá una nueva ventana con un sencillo editor de texto donde podremos escribir el programa, guardarlo y posteriormente ejecutarlo.

También se puede crear y ejecutar programas desde el terminal, en este caso lo que deberemos hacer es crear un archivo de texto con cualquier editor de texto (e.g. nano) con el que editar el código Python. A continuación se muestran dos imágenes de cómo se ejecutan dos simples aplicaciones como, “Hello, World!” tanto con el IDLE de Python como por terminal.

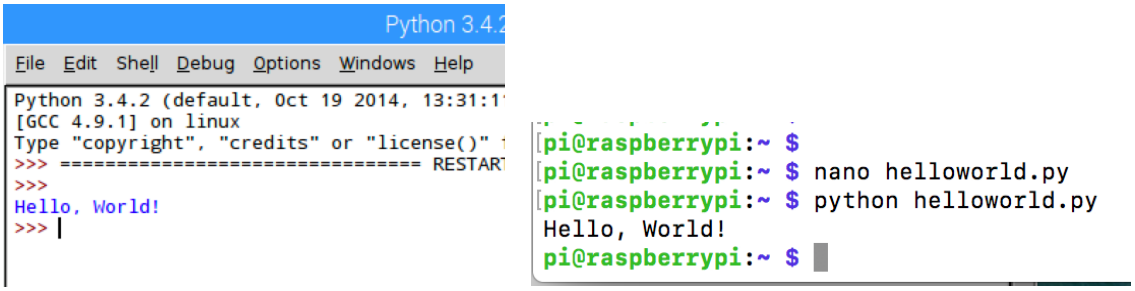


Fig. 3.4 Dos maneras de ejecutar una aplicación en Python (IDLE y terminal)

3.2.3 Instalación de la Pi Camera

Antes de empezar con el montaje de la cámara trampa deberemos instalar la cámara y comprobar su funcionamiento con algún programa simple.

Lo primero a realizar es comprobar que tenemos habilitada la interfaz de la cámara (figura 3.1). Una vez habilitada la interfaz realizaremos la conexión física al puerto CSI de la Raspberry (Sección 2.2.1), tal y como muestra la siguiente imagen.

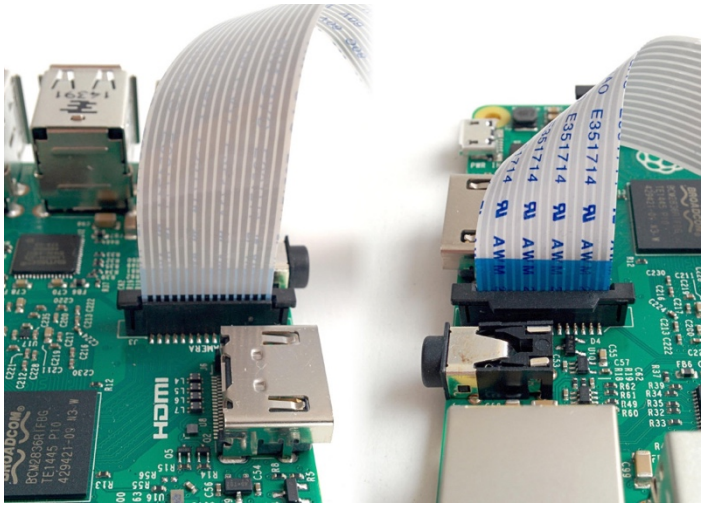


Fig. 3.5 Conexión física del módulo cámara con la Raspberry

Hay muchas maneras de utilizar la cámara, pero en este proyecto se gestionará mediante scripts Python, por lo que deberemos instalar las librerías de la cámara en Python con el siguiente comando:

```
pi@raspberrypi:~ $  sudo apt-get install python-picamera
```

Una vez instalada la librería, será momento de probarla realizando una simple aplicación para familiarizarse con el entorno de trabajo. Para ello podemos abrir un nuevo archivo con el IDLE de Python 3 al que nombraremos *Camera.py* y le añadiremos el siguiente código.

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()
camera.rotation = 180
camera.start_preview()
sleep(5)
camera.capture ('/home/pi/Desktop/pruebas/image.jpg')
camera.stop_preview()
```

En el código anterior podemos observar como en las primeras dos líneas se importan las librerías que previamente hemos instalado, en la tercera línea creamos un objeto PiCamera. Este tipo de objeto permite la manipulación de la cámara. En la quinta línea nos muestra lo que ve la cámara en la pantalla que

tengamos conectada a la Raspberry. Esta previsualización (*preview*) de la imagen solo funciona con un monitor conectado, mientras estemos en remoto no podremos ver la previsualización. La siguiente línea es para poder poner un tiempo de espera. Es importante poner una espera de por lo menos 2 segundos antes de capturar, para dar al sensor de la cámara tiempo para establecer sus niveles de luz. Posteriormente, se llama al comando para realizar la captura e indicamos entre paréntesis la ruta en la que queremos guardar la imagen. Finalmente se termina la previsualización de la cámara. Con esta pequeña aplicación comprobamos el fácil funcionamiento de la cámara.

CAPÍTULO 4. DESARROLLO DE LA SOLUCIÓN

La finalidad de este capítulo es describir con detalle la implementación de las soluciones propuestas para el desarrollo del proyecto. La soluciones a implementar serán desarrolladas con diferentes tipos de detectores de movimiento. El primero será con un sensor infrarrojo, el segundo con detección por software y finalmente uno con un sensor de ultrasonidos.

Se implementará una mejora para poder enviar los archivos automáticamente tras realizar la fotografía.

4.1. Escenario 1: Cámara trampa con sensor PIR

Nuestro primer escenario para recrear el funcionamiento de una cámara trampa será utilizando el PIR sensor, como sensor de movimiento. Grandes fabricantes de cámaras trampa utilizan este tipo de sensores [11]. El esquema a realizar es simple, lo primero será conectar la Pi Camera a la Raspberry como hemos comentado en apartados anteriores y posteriormente realizar el cableado entre el sensor y la Raspberry, que será el siguiente:

- Conectamos pin VCC del sensor a la clavija de 5V (pin 2) de la Raspberry, proporcionando energía al sensor.

- Conectamos el pin GND a la clavija de tierra (pin 6) de la Raspberry, para completar el circuito.

- Conectamos el pin OUT al GPIO4 (pin 7) de la Raspberry, así el sensor producirá un voltaje cuando se detecte un movimiento que puede ser recibido en la Raspberry.

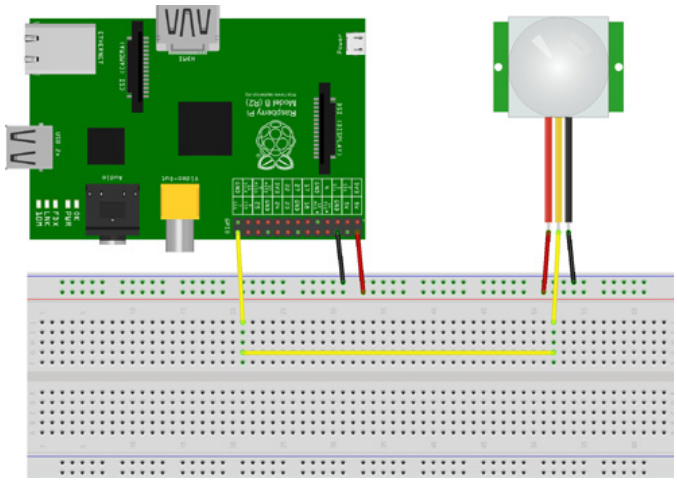


Fig. 4.1 Esquema de conexión del sensor infrarrojo

Una vez realizado el cableado, será el momento de comprobar que tenemos el sensor correctamente conectado. Para ello crearemos un simple programa en el que utilizaremos la librería GPIO Zero, que viene instalada en Raspbian por defecto. Dicha librería fue creada para facilitar el trabajo, al utilizar los pines y hacerla mucho más amigable para el usuario. La GPIO Zero, proporciona la posibilidad de importar los periféricos conectados en los pines del GPIO. Esta librería permite importar los siguientes periféricos:

- LEDs
- Timbre
- Motor
- Pulsador
- Sensor de movimiento
- Sensor de luminosidad

El programa que utilizaremos para comprobar el funcionamiento del sensor es el siguiente:

```
from gpiozero import MotionSensor

pir = MotionSensor(4)
while True:
    if pir.motion_detected:
        print("Motion detected!")
```

Esta simple aplicación nos mostrará en pantalla el mensaje "*Motion detected*" cada vez que el sensor detecte un movimiento, comprobando que el sensor está correctamente conectado.

El próximo paso será adaptar el programa para que cada vez que el sensor detecte movimiento realice una foto, para ello realizaremos una adaptación entre esta última aplicación y la que vimos previamente para realizar fotos.

```
from gpiozero import MotionSensor
from picamera import PiCamera
import time

camera = PiCamera()
pir = MotionSensor(4)
counter= 1
while True:
    pir.wait_for_motion()
    camera.rotation=180
    camera.start_preview()
    camera.capture ('/home/pi/Desktop/pruebas/image%s.jpg' % counter)
    counter = counter + 1
    print ('Motion detected!')
    pir.wait_for_no_motion()
```

```
camera.stop_preview()
time.sleep(3)
```

En este programa podemos ver cómo en la primera parte importamos las librerías que utilizaremos, la siguiente parte crea los objetos que controlan la cámara y el sensor y, finalmente, el cuerpo de la aplicación que cada vez que detecta movimiento la cámara hace una foto en la carpeta que hemos creado previamente.

4.2. Escenario 2: Cámara trampa con procesado de imagen

El siguiente escenario se realizará únicamente con la Pi Camera y el detector de movimiento se ejecutará mediante software. Hay diferentes maneras de realizar la detección de movimiento mediante software, en concreto se han probado 2 maneras diferentes:

- OpenCV es una biblioteca libre en la que hay una gran cantidad de aplicaciones, desde sistemas de detección de movimiento hasta aplicaciones de control de procesos. El principal problema que se ha encontrado de este método es que no acepta la Pi Camera y se debería realizar con una de las cámaras compatibles (Desde una GoPro hasta una Réflex).

- Aplicación Python realizada por terceros: Hay aplicaciones ya realizadas por terceros con la que es posible realizar fotografías y realizar una sencilla detección de movimiento por software. La ventaja de este sistema es que permite la utilización de la Pi Camera, además de poder configurar la aplicación a nuestra medida para que tenga una sensibilidad u otra.

Finalmente se ha decidido implementar la aplicación realizada por terceros por las ventajas de las que disponíamos a la hora de configurar la detección y además de ser compatible con nuestra cámara. En concreto, se ha decidido utilizar una versión de Claude Pageau [10]. Con esta aplicación podremos optimizar algunas variables para tenerla adaptada mejor a nuestra aplicación.

Entre ellas las más significativas son las relacionadas con la captura de la imagen (directorio y nombre), el tamaño de la imagen (número de píxeles horizontales y verticales), el volteo de la imagen y, sobretodo, la sensibilidad o umbral que queremos que tenga nuestra aplicación para detectar movimiento entre una imagen y otra.

- Threshold: Este valor indica cuánto debe cambiar un píxel antes de que el programa se dé cuenta. El valor varía entre 1 y 254 (254 sería un cambio de negro a blanco).

- Sensitivity: Este valor nos indica el número de píxeles que deben cambiar antes de que se detecte movimiento. Como más alto es el valor menos sensibilidad.

Tras entender la labor de estas dos últimas variables, ya podemos ver que para que se detecte movimiento se debe tener en cuenta el cambio de tonalidad del píxel y del número de píxeles que cambian.

En el siguiente párrafo se muestra un trozo del código de la aplicación que podemos encontrar en el Anexo 1.

```
def scanMotion(width, height, daymode):
    motionFound = False
    data1 = takeMotionImage(width, height, daymode)
    while not motionFound:
        data2 = takeMotionImage(width, height, daymode)
        diffCount = 0L;
        for w in range(0, width):
            for h in range(0, height):
                #Obtiene la diferencia del pixel
                diff = abs(int(data1[h][w][1]) - int(data2[h][w][1]))
                if  diff > threshold:
                    diffCount += 1
            if diffCount > sensitivity:
                break;
        if diffCount > sensitivity:
            motionFound = True
        else:
            data2 = data1
    return motionFound
```

En esta parte del código podemos observar como la aplicación utiliza los dos valores preconfigurados para poder detectar movimiento. De esta manera con lo que se ha comentado en párrafos anteriores se puede ajustar la aplicación para configurar la sensibilidad de detección.

4.3. Escenario 3: Cámara trampa con sensor ultrasónico

En este último escenario utilizaremos el sensor ultrasónico para la detección de movimiento. Este sensor suele utilizarse para calcular la distancia de los objetos que tiene delante, pero aquí se utilizara para detectar movimiento. En primero lugar se realizará el esquema de conexión entre la Raspberry y el sensor, que será el siguiente:

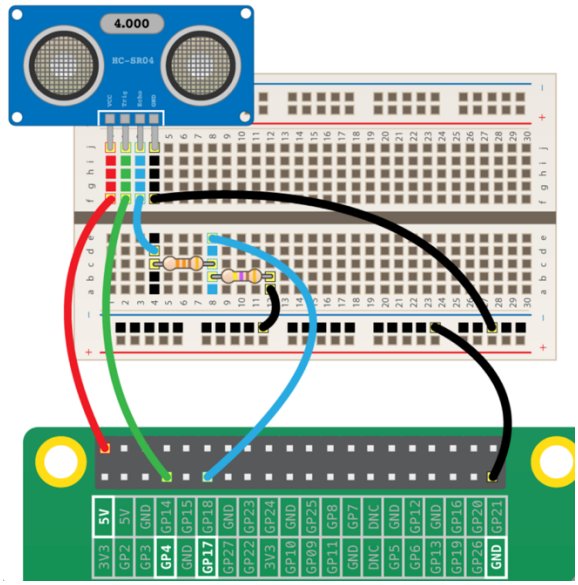
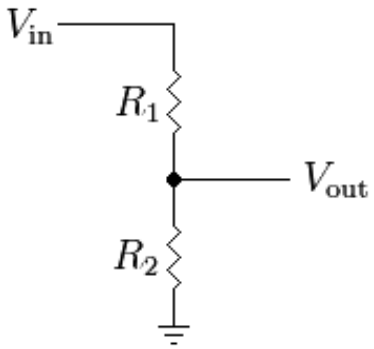


Fig. 4.2 Esquema de conexión del sensor de ultrasonidos

En la imagen anterior se puede observar que, tal y como se ha comentado en apartados anteriores, se aconseja el uso de un divisor de tensión para ajustar el voltaje que envía el sensor a la Raspberry, ya que el sensor trabaja a 5V y nuestra Raspberry a 3V. En nuestro caso, para realizar el divisor de tensión hemos utilizado una resistencia de 330Ω y otra de 470Ω, las cuales las hemos obtenido tras realizar un simple cálculo que se utiliza para los divisores de tensión. El calculo es el siguiente:



$$V_{out} = \frac{R_2}{R_2 + R_1} \cdot V_{in} = \frac{470}{470 + 330} \cdot 5V = 3V$$

Fig. 4.3 Esquema del divisor de tensión y su resolución

El siguiente paso a seguir será realizar la conexión de los pines del sensor hacia la Raspberry. La conexión la realizaremos de la siguiente manera:

- Conectamos el pin VCC del sensor a la clavija de 5V (pin 4) de la Raspberry.
- Conectamos el pin GND a la clavija de tierra (pin 39) de la Raspberry.
- Conectamos el pin Trig al GPIO23 (pin 16) de la Raspberry.

- Conectamos el pin Echo al GPIO24 (pin 18) de la Raspberry.

Para comprobar que el sensor funciona correctamente crearemos un programa simple con el cual podemos establecer la distancia a la que está el objeto del sensor. Como con el sensor infrarrojo se utilizará la librería GPIO Zero para cargar los pines del sensor. El código será el siguiente:

```
from gpiozero import DistanceSensor
import time
ultrasonic = DistanceSensor(echo=24, trigger=23)
ultrasonic.distance
while True:
    print(ultrasonic.distance)
    time.sleep (2)
```

A través de este programa se envía una señal por el pin TRIG y cuando el rebote llega al pin ECHO nos aparece la distancia a la que está el objeto.

Además de poder ver el valor de la distancia, también se puede hacer que el sensor envíe una señal por algún pin cuando un objeto está dentro o fuera de un cierto rango de detección. Esta característica la utilizaremos para disparar la cámara cuando el objeto esté dentro de rango. El programa que emplearemos para capturar la imagen cuando un objeto esté dentro de rango es el siguiente:

```
from gpiozero import DistanceSensor
from picamera import PiCamera
import time
ultrasonic = DistanceSensor(echo=24, trigger=23,threshold_distance=0.9)

ultrasonic.distance
camera = PiCamera()
counter=0

while True:
          ultrasonic.wait_for_in_range()
        print ("Dentro del rango")
        camera.rotation=180
        camera.start_preview()
        camera.capture ('/home/pi/Desktop/pruebas/image%s.jpg' % counter)
        counter = counter + 1
        camera.stop_preview()
        ultrasonic.wait_for_out_of_range()
        print ("Fuera de rango")
        time.sleep (1)
```

Con este programa podemos observar cómo la aplicación está en espera hasta que un objeto entra dentro del rango del sensor, en cuyo caso, manda la orden de realizar una foto. Una vez sale de rango, manda un mensaje y vuelve a esperar a que aparezca otro objeto dentro de su rango. En este programa, el valor que podemos modificar es `threshold_distance` con el cual podremos configurar la sensibilidad del sensor. Este valor puede ir de 0 a 1; cuanto más alto más rango de detección.

4.4. Mejoras

Como mejora para el proyecto se ha decidido realizar la transmisión de las fotografías tomadas a un servidor externo para así no acumularlas en la Raspberry y evitar que la tarjeta se llene. Como ejemplo podemos hablar de cuando una cámara on-ride envía una fotografía tomada de una atracción al quiosco dónde se muestran e imprimen dichas imágenes, para su posterior venta.

Para enviar las imágenes de manera segura, se propone el uso del protocolo SFTP, de manera que realizaremos un script para enviar automáticamente las imágenes a medida que las vamos realizando.

En primer lugar se configurara el cliente y servidor para utilizar las claves públicas y privadas, de manera que no tengamos que autentificar manualmente el acceso al servidor cada vez que queramos enviar las imágenes.

La principal razón para usar llaves privadas con ssh y ftp es automatizar la copia de archivos entre la Raspberry y el servidor a medida que se van realizando nuevas imágenes. Si lo realizáramos simplemente con FTP tenemos el riesgo de que alguna tercera persona pueda interceptar la transmisión, por lo que no es indicado realizar una conexión no segura.

Para crear estas llaves en el equipo del cliente (la Raspberry), deberemos generar las llaves y el usuario que se conectará al servidor. Como ejemplo usaremos el usuario 'pi' en el equipo cliente 'Raspberry' y lo realizaremos utilizando el comando `ssh-keygen`. El comando a realizar será el siguiente:

```
pi@raspberrypi:~ $ ssh-keygen -b 1024 -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_dsa):
Enter passphrase (empty for no passpharase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_dsa.
Your public key has been saved in /home/pi/.ssh/id_dsa.pub.
The key fingerprint is:
2f:08:72:b0:a0:cc:2e:59:92:85:9e:1f:90:59:be:3e pi@raspberry
```

La opción –t dsa indica que utilizaremos el algoritmo DSA (Digital Signature Algorithm) como medio de encriptación. El programa nos permite elegir el directorio donde guardar la llave pública y privada que lo dejaremos en el directorio por defecto. A continuación se solicita una contraseña que podemos dejar en blanco. El próximo paso será copiar el certificado o llave publica en el servidor Linux.

El siguiente paso a realizar en el servidor es crear la misma carpeta que se ha creado en el cliente.

```
pi@servidor:~ $ pwd
/home/pi
```

```
pi@servidor:~ $ mkdir  .ssh; chmod 700 .ssh
pi@servidor:~ $ cd .ssh
pi@servidor:~/.ssh $ touch authorized_keys; chmod 600 authorized_keys
```

Con estos comandos hemos creado el directorio '.ssh' con los permisos necesarios. Dentro de la carpeta, se ha creado el archivo 'authorized_keys' que tendrá permisos de lectura y escritura y que estará vacío. Posteriormente, añadiremos el contenido 'id_dsa.pub'.

El próximo paso será copiar desde el cliente el contenido de la llave a la carpeta '.ssh' del servidor.

```
pi@raspberrypi:~/.ssh $ cat id_dsa.pub | ssh pi@servidor "cat >> ~/.ssh/authorized_keys"
```

Una vez el servidor obtiene el certificado autorizado para hacer login, podemos comprobar que no se necesita introducir la contraseña cuando utilicemos el protocolo SFTP.

A continuación, realizaremos un script para que, una vez realizada la fotografía, la Raspberry envíe la imagen al servidor que previamente hemos configurado con las llaves. Esta transferencia se realizará con el protocolo SFTP.

```
#!/bin/bash
echo "Comienzo del Script..."
sftp -i id_server_dsa pi@192.168.0.166 <<EOF #Realiza la conexión mediante sftp
cd /home/pi/Desktop/probandoscript #Ruta destino de las imagenes
lcd /home/pi/Desktop/pruebas #Ruta local de las imagenes
mput * #Copia los archivos de la carpeta local
bye
EOF
```

Una vez creado el script, en un archivo de texto al que daremos el formato Shell Script. A partir de este punto se tendrá que modificar los códigos de los escenarios que se han creado previamente añadiendo unas líneas a final de cada código, para poder llamar a dicho script cada vez que se realice una nueva fotografía. En el siguiente ejemplo aparece el código del primer escenario al que añadimos las líneas necesarias para realizar la conexión SFTP y borrar las fotografías antiguas.

```
from gpiozero import MotionSensor
from picamera import PiCamera
import time
import os
import subprocess

camera = PiCamera()
pir = MotionSensor(4)
counter= 0
while True:
pir.wait_for_motion()
    camera.rotation=180
    camera.start_preview()
    camera.capture ('/home/pi/Desktop/pruebas/image%s.jpg' % counter)
```

```
    counter = counter + 1
    print ('Motion detected!')
    pir.wait_for_no_motion()
    camera.stop_preview()
    if counter > 1:
        count = counter - 2
        os.remove('/home/pi/Desktop/pruebas/image%s.jpg'%count)
#borra las imágenes previas
    time.sleep(2)
    os.system ("./sftp") #Llama al script para ejecutar el SFTP
```

Las líneas destacadas en amarillo son las que deberemos añadir para poder realizar la llamada al script, que es la última línea de todas. Las otras líneas añadidas son las que borran las imágenes tomadas previamente, para así evitar que se nos cargue en exceso el almacenamiento de la tarjeta microSD de la Raspberry.

CAPÍTULO 5. EVALUACIÓN DE LAS DIFERENTES PROPUESTAS

En este capítulo trataremos de evaluar los diferentes escenarios y ver su funcionamiento en diferentes situaciones. Además, se propone un presupuesto con los materiales y horas invertidas.

Finalmente, se ha realizado un apartado describiendo el correcto funcionamiento del proyecto.

5.1. Pruebas realizadas

Tras montar los escenarios y realizar las aplicaciones de cada una de ellas, el próximo paso es utilizarlos y realizar las pruebas pertinentes para ver cuál puede ser la mejor opción a la hora de realizar un montaje definitivo.

Vamos a realizar una serie de pruebas a diferentes velocidades para ver si los sensores reaccionan correctamente con la variación de velocidades. Además, se realizará una prueba para ver cuál es el grado de detección de cada sensor. Todas las pruebas se van a realizar 20 veces por escenario para poder ofrecer un resultado fiable.

Las pruebas que se realicen a un velocidad de 5 km/h serán realizadas con una persona que se desplaza sobre el plano horizontal de la cámara. Las que se realicen a una velocidad aproximada de 20 km/h se han realizado con una persona en bicicleta que se desplaza sobre el plano horizontal del objeto. Todas las pruebas se han realizado con buenas condiciones de luz.

La primera prueba realizada es con un sujeto que se mueve a una velocidad aproximada de 5 km/h y tiene visibilidad directa entre el dispositivo y el objeto como se muestra en la siguiente imagen.

Fig. 5.1 Primera prueba a 5 km/h y visibilidad directa

Los resultados presentados son los siguientes:

Tabla 5.1. Prueba realizada a 5 km/h y visibilidad directa

Directa – 5 km/h	PIR	Software	Ultra
<1m	100%	100%	95%
3m	100%	100%	40%
7m	95%	100%	0%

La próxima prueba realizada será como la anterior, pero con el sujeto moviéndose a mayor velocidad. En este caso irá a una velocidad aproximada de 20 km/h . La imagen siguiente muestra cómo se ejecuta la prueba:

Fig. 5.2 Segunda prueba a 20 km/h y visibilidad directa

Los resultados son los siguientes:

Tabla 5.2. Prueba realizada a 20 km/h y visibilidad directa

Directa – 20 km/h	PIR	Software	Ultra
<1m	100%	40%	30%
3m	100%	85%	10%
7m	90%	100%	0%

La próxima prueba a realizar consiste en buscar cuál es el ángulo de detección en el cual se activa el dispositivo y se realiza la fotografía habiendo una distancia de 5 metros en los dos primeros escenarios y de 3 metros en el tercero entre el sujeto y el dispositivo. La siguiente imagen describe como se realiza el proceso:

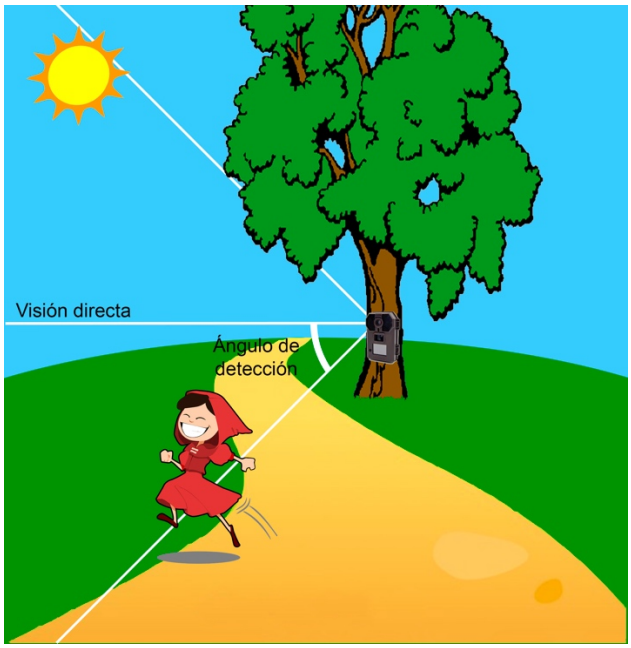


Fig. 5.3 Tercera prueba para saber ángulo de detección

Los resultados son los siguientes:

Tabla 5.3. Prueba ángulo de detección

	PIR	Software	Ultra
Ángulo	110º	80º	12º

La última prueba que se ha realizado consiste en comprobar la distancia máxima que puede haber entre el dispositivo y un objeto con visión directa para que se realice la detección. Los resultados han sido los siguientes:

Tabla 5.4. Prueba distancia máxima

	PIR	Software	Ultra
Ángulo	7 metros	>15 metros	3 metros

A continuación, analizaremos todas las pruebas realizadas. En la primera prueba se ha tratado de comprobar el funcionamiento de los escenarios a diferentes distancias a la velocidad de un peatón. Tanto el sensor infrarrojo como el procesado de la imagen se han comportado de una manera muy efectiva, presentando buenos resultados. Mientras que el sensor ultrasonidos a medida que se separa del sensor pierde fiabilidad y presenta más errores a mayor distancia.

En la segunda prueba se trata de comprobar el funcionamiento de los escenarios con objetos a mayor velocidad, en este caso el objeto va en bicicleta. Aquí los

escenarios, se han comportado de manera menos eficiente, siendo el sensor infrarrojo el que mayor tasa de éxito ha presentado. El procesado de imagen, a medida que se separa el objeto del dispositivo presenta una mayor eficacia, esto es debido a que cuanto mayor es la distancia, con más facilidad se producen cambios detectables en la imagen. Finalmente, se ha podido observar como el sensor de ultrasonidos no ha funcionado como se esperaba, siendo el peor en esta prueba.

Seguidamente, se ha tratado de comprobar el ángulo de visión que tiene cada escenario. El sensor infrarrojo y el procesado de imagen han presentado un buen rango de visión, mientras que el sensor de ultrasonidos únicamente ha podido detectar en un ángulo de 12º.

En la distancia máxima de detección, el sensor infrarrojo presenta el mayor valor de los medidos, mientras que el ultrasónico tiene un valor insuficiente para la realización de una cámara on-ride, pero puede tener buenos valores para la observación de fauna salvaje. Se ha podido observar que existe variabilidad en el procesado de imagen en relación a la distancia. Esta variación está relacionada a la distancia que presenta nuestra cámara al fondo del paisaje.

Tras el uso de los anteriores escenarios y de las pruebas realizadas, las características de cada escenario se resumen a continuación:

- Sensor Infrarrojo: Ha presentado un alto porcentaje de aciertos y un buen funcionamiento en las pruebas de campo tanto a diferentes distancias como velocidades. El que tenga un ángulo de visión mayor y que pueda colocarse el sensor y la cámara en posiciones distintas hace que tenga una alta capacidad de adaptación, además de tener una fácil configuración.

- Procesado de imagen: Este escenario ha dado buenos resultados a bajas velocidades, pero cuando se ha aumentado la velocidad, solo ha presentado buenos resultados a medida que se alejaba el objeto de la cámara. Este se puede solucionar ajustando el tiempo entre las imágenes capturadas, de manera que a cortas distancias no presente errores. Otro problema que ha tenido este sensor es que ha presentado detecciones erróneas cuando ha habido un cambio en el paisaje (e.g movimiento de las plantas por el viento). Este escenario presenta una configuración muy cómoda, ya que únicamente hemos de conectar la cámara. La parte negativa es que hay muchas variables a tener en cuenta para adaptar el sistema a cada escenario.

- Sensor Ultrasónico: Presenta una peor detección que el resto de configuraciones. A medida que se aleja el objeto del sensor y a medida que se aumenta la velocidad empeora su sensibilidad. También nos ha presentado falsas detecciones en algunas ocasiones. Este escenario estaría más enfocado a la observación de fauna a poca distancia, ya que su mayor directividad y menor rango evita falsos positivos y permite la realización de buenas capturas a corta distancia. En definitiva, sería el último escenario a elegir para nuestro proyecto. Ya que a pesar de los

pobres resultados, se ha de montar un pequeño circuito electrónico que aumenta su complicación.

A continuación se muestra una tabla comparativa resumida de las tres soluciones estudiadas:

Tabla 5.5. Tabla comparativa de los diferentes escenarios

	Ventajas	**Desventajas**
Escenario 1	• Ajustar sensibilidad • Bajo consumo del sensor • Alto ángulo de detección • Buen comportamiento a diferentes velocidades • Buena detección hasta los 7 metros	• Posibilidad de que el sol caliente algún objeto cercano y haga saltar el sensor
Escenario 2	• Amplia posibilidad de configuración • Distancia máxima muy alta • Bien a altas velocidades • Fácil montaje	• Posibles detecciones erróneas • Alto consumo • Alta complejidad del código
Escenario 3	• Bien para distancias cortas • Buen funcionamiento para objetos lentos	• Poco ángulo de visión • Peor funcionamiento a mayor distancia • Mal funcionamiento a altas velocidades

Tras revisar los tres escenarios, se ha decidido que el mejor escenario para realizar la cámara on-ride sería con el sensor infrarrojo. Por las razones que se han visto a lo largo de este apartado se ha implementado este escenario como versión final.

5.2. Valoración económica

Tras haber escogido la configuración más apropiada, el siguiente paso será realizar una valoración económica para poder efectuar un presupuesto del montaje final.

- Raspberry PI 3 Modelo B: 35€

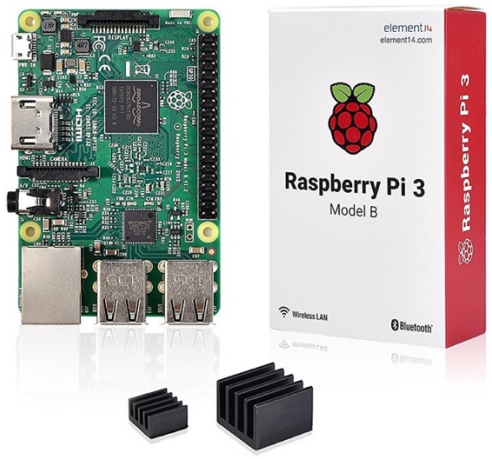


Fig. 5.4 Raspberry Pi 3 Modelo B

- Pi Camera v2: 28€

Fig. 5.5 Módulo Cámara v2

- Carcasa: 19€

Fig. 5.6 Carcasa para la cámara trampa

- PIR Sensor: 6€

Fig. 5.7 Sensor infrarrojo

- MicroSD: 7€

Fig. 5.8 Tarjeta microSD

- Fuente de alimentación: 11€

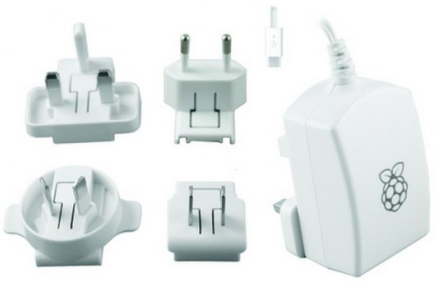


Fig. 5.9 Fuente de alimentación

Para la elección de la carcasa se ha escogido un modelo de exterior que no es resistente al agua. Si por algún motivo, el dispositivo debiera mojarse se debería adquirir una carcasa resistente al agua y realizar los cambios necesarios para poder introducir el sensor y la cámara.

El precio final de los materiales necesarios para poder realizar correctamente el dispositivo será el siguiente:

Tabla 5.6. Tabla de precios del proyecto

Material	Precio
Raspberry Pi 3 Modelo 3	35€
Pi Camera v2	28€
Carcasa	19€
PIR Sensor	6€
MicroSD	7€
Fuente alimentación	11€
Total	106€

Una vez conocido el precio de los materiales, deberemos saber el coste de personal para la realización del proyecto. Contando con un desarrollador con suficiente conocimiento sobre las diferentes tecnologías involucradas, se podría realizar el proyecto en unas 280h (20h semanales durante 14 semanas). Notar que este cálculo no incluye el tiempo necesario para su instalación y calibración en el escenario elegido. El precio por mano de obra, a unos 8€/h, hacen un total de 2240€. El precio final seria el siguiente:

Tabla 5.7. Tabla de precio total del proyecto

Concepto	Precio
Material	106€
Mano de obra	2240€
Total	2346€

5.3. Funcionamiento

En este apartado vamos a hacer una simulación del funcionamiento del proyecto desde cero.

1. Conectamos la Raspberry a la corriente y esperamos a que se conecte a internet.

```
pi@raspberrypi:~ $ ping 192.168.0.154
PING 192.168.0.154 (192.168.0.154) 56(84) bytes of data.
64 bytes from 192.168.0.154: icmp_seq=1 ttl=64 time=0.113 ms
64 bytes from 192.168.0.154: icmp_seq=2 ttl=64 time=0.090 ms
64 bytes from 192.168.0.154: icmp_seq=3 ttl=64 time=0.102 ms
```

2. Nos conectamos a la Raspberry por terminal y activamos el VNC server para poder conectarnos con el cliente.

```
MacBook-Pro-de-Aaron:~ aaronvinentpons$ ssh pi@192.168.0.154
The authenticity of host '192.168.0.154 (192.168.0.154)' can't be established.
ECDSA key fingerprint is SHA256:hdDnMLWi/V8G+k7IzVtv/IyRTQECiV8MXpWStmLa3H0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.154' (ECDSA) to the list of known hosts.
pi@192.168.0.154's password:
pi@raspberrypi:~ $ tightvncserver

New 'X' desktop is raspberrypi:1

Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log
```

3. Una vez realizada la conexión remota conectaremos la Raspberry al servidor para comprobar que la carpeta de destino esta vacía. Además abrimos la aplicación en el IDLE de Python y abrimos el directorio donde se almacenan las fotografías.

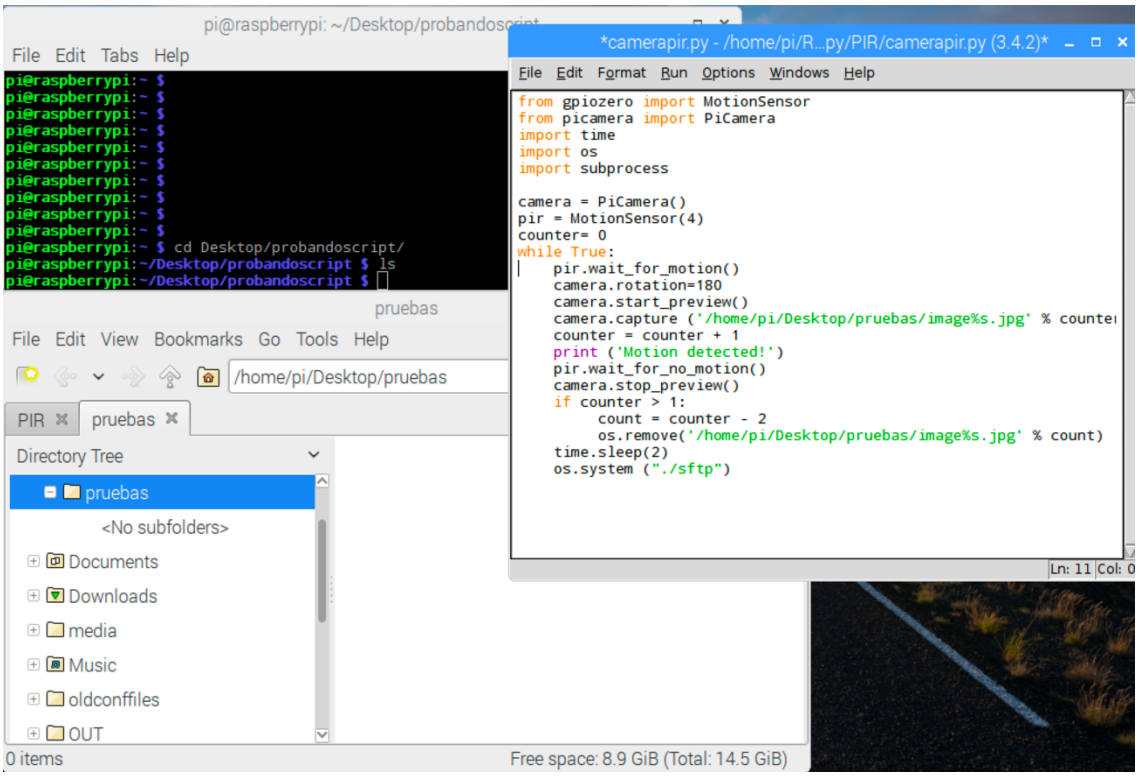


Fig. 5.10 Escritorio de la Raspberry con el código del proyecto

4. Con F5 iniciamos la aplicación Python y esperamos ver cómo se va realizando el proceso.

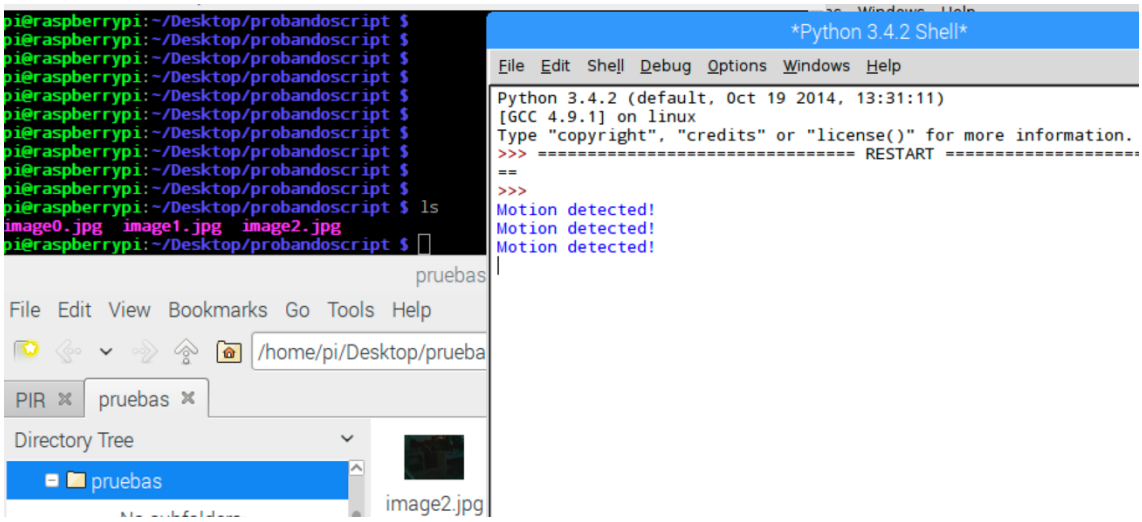


Fig. 5.11 Funcionamiento del proyecto

A continuación, se van a mostrar algunas imágenes que se han obtenido con las pruebas realizadas anteriormente.

Fig. 5.12 Imagen de muestra peatón

Fig. 5.13 Imagen de muestra peatón

Fig. 5.14 Imagen de muestra de un corredor

Como se puede observar en las imágenes anteriores, la calidad de la cámara no es la deseada cuando el objeto se mueve a cierta velocidad. En la última imagen se puede comprobar que a altas velocidades no se distingue bien el objeto en cuestión. En conclusión, sería aconsejable utilizar otro tipo de cámara para realizar la captura de las imágenes, ya que la cámara de la Raspberry es muy limitada a la hora de capturar imágenes en movimiento.
A lo largo de este apartado se ha podido observar el correcto funcionamiento del dispositivo y cómo a medida que vamos realizando fotografías se van enviando al servidor y las imágenes previas se borran.

CAPÍTULO 6. CONCLUSIONES Y FUTURAS MEJORAS

El objetivo de este trabajo es la realización de una cámara trampa, en concreto una cámara on-ride, para poder ofrecer una alternativa de bajo coste a los sistemas on-ride del mercado. Para ello se han probado distintas tecnologías y se ha aprovechado el potencial de la Raspberry Pi.

Para realizar este último apartado se va a dar una versión más subjetiva sobre el proyecto realizado y se va a comentar el veredicto que se ha obtenido tras las pruebas realizadas.

Se han implementado diferentes soluciones para el reto que se proponía al principio, esto es, una cámara on-ride de bajo coste. Las diferentes soluciones se diferencian según el método de detección de movimiento. Después, se ha realizado un estudio de los resultados ofrecidos y, finalmente, se ha seleccionado la mejor opción.

Una vez finalizado el proyecto, podemos decir que para una cámara on-ride las dos primeras soluciones (detector infrarrojo y detector por procesado de imagen) han alcanzado los objetivos esperados presentando un alto porcentaje de detecciones correctas y un bajo índice de errores. En el caso de implementar el sistema, se decidiría utilizar el sistema de detección mediante un sensor infrarrojos. Como se ha comentado en otros apartados, el sensor ultrasónico presenta un mejor uso para detecciones directas y con poco movimiento, como puede ser la observación de la fauna.

Por último, un punto a tener en cuenta, es que la cámara que se comercializa y está especialmente diseñada para su funcionamiento con la Raspberry Pi es muy limitada en cuanto a calidad de la imagen, lo que puede provocar que no todas las fotografías queden de la manera deseada. Están más pensadas para sistemas de vigilancia o para cámaras trampa en la que el objeto se mueve a menor velocidad que para las cámaras on-ride. Por ello, se recomienda que para futuras mejoras, si se ha de realizar este tipo de dispositivo, se utilice una cámara réflex.

5.1. Futuras mejoras

Una vez finalizado el proyecto y testeado el funcionamiento de la cámara on-ride, hay algunos puntos en los que se podrían añadir cambios para mejorar el rendimiento final. En los próximos párrafos se enumeran los problemas que se han encontrado y propuestas para solucionarlos.

- Calidad de la imagen: después de la realización de las pruebas se ha podido observar que la calidad de las imágenes realizadas no cumple con los requisitos esperados. A mayor velocidad, menor calidad de la imagen. Para ello se recomienda utilizar una cámara réflex o DSLR. Este tipo de cámara ofrecerá un aumento significativo de la calidad de imagen a

diferentes velocidades. Para realizar el cambio de cámara, primero debemos comprobar que es compatible con la Raspberry Pi [5] mediante conexión USB. Tras ello se deberá realizar un pequeño cambio en el programa para cuando se detecte movimiento se realice la fotografía. Esto será posible utilizando el programa gPhoto2 (un software open source para el control remoto de cámaras) [12].

- Sistema de comunicación inalámbrico: En el proyecto se ha escogido el sistema de comunicación con el Wi-Fi integrado de la Raspberry Pi, por su facilidad de implementación. Para futuras mejoras se debería realizar la conexión Wi-FI mediante un adaptador USB que soporte el estándar 802.11ac, que mejorará considerablemente la velocidad y alcance de la transmisión.

- Inclusión de un flash: En el caso de que la luz no sea suficiente, seria conveniente incorporar un flash al sistema. Lo ideal seria utilizar un flash de LEDs para el ahorro de energía

BIBLIOGRAFÍA

[1] Trekkinn. *Bushnell 8 Mp Trophy Cam HD Wireless* [en línea]. [consulta: 16 de julio de 2017]. Disponible en <https://www.trekkinn.com/montana/bushnell-8-mp-trophy-cam-hd-wireless/1299920/p?utm_source=google_products&utm_medium=merchant&id_producte=1630924&country=es&gclid=Cj0KEQjwwLHLBRDEq9DQxK2I_p8BEiQA3UDVDqTFmgH7gW1ECbdlga7Lvw-CL1Sv8UdH4ZlL3oDmIw8aAjCG8P8HAQ&gclsrc=aw.ds>

[2] Generalitat de Catalunya. [en línea]. [consulta: 16 de julio de 2017]. Disponible en <https://contractaciopublica.gencat.cat/ecofin_pscp/AppJava/portalfileretrieving.pscp?reqCode=retrieveFile&docHash=1a6456e842813f1b132cb445b3d0d916&fileId=18602709&capId=16219628&idTS=18602392>

[3] Wikimedia Commons: Meadows SM 1 coyote [en línea].[consulta: 12 de julio de 2017]. Disponible en <https://commons.wikimedia.org/wiki/File:Sky_Meadows_SM_1_coyote_(16132074235).jpg>

[4] Carlos. (27 de abril de 2008). Splash Mountain: Posing for the On-Ride Photo. [Archivo fotográfico]. Recuperado de <https://www.flickr.com/photos/armadillo444/2500049533>

[5]Raspberry para torpes. *Raspberry Pi 3 thermal image* [en línea]. Sobre la Raspberry Pi 3, 14 de marzo de 2016. [consulta: 19 de junio de 2017]. Disponible en ≤https://i1.wp.com/raspberryparatorpes.net/wp-content/uploads/2016/03/raspberry-pi-3-thermal-image-.png>

[6]eLinux. *RPi VerifiedPeripherals* [en línea]. [ultima revisión:5 de julio de 2017]. [consulta: febrero de 2017]. Disponible en <http://elinux.org/RPi_VerifiedPeripherals>

[7]Punto Flotante. *Manual del usuario sensor de movimiento PIR HC-SR501* [en línea]. [consulta: febrero de 2017]. Disponible en <http://www.puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf>

[8] Raspberry Pi Foundation. *Installing operating system images using Windows* [en línea]. [consulta: 12 de julio de 2017]. Disponible en <https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

[9] Raspberry Pi Foundation. *Installing operating system images on Linux* [en línea]. [consulta: 12 de julio de 2017]. Disponible en <https://www.raspberrypi.org/documentation/installation/installing-images/linux.md>

[10] Real VNC. *Download VNC Viewer to the device to control from* [en línea]. [consulta: febrero de 2017]. Disponible en <https://www.realvnc.com/download/viewer/>

[11] Browning Trail Cameras. *How a Motion/Heat (PIR) Sensor Works, 17 de Marzo de 2017. [consulta: Junio de 2017]. Disponible en* <https://browningtrailcameras.zendesk.com/hc/en-us/articles/216821947-How-a-Motion-Heat-PIR-Sensor-Works>

[12] GitHub. *PiCamera Motion.* A: Claude Pageau [en línea]. 8 de juliode 2015. [consulta: Febrero de 2017]. Disponible en <https://github.com/pageauc/picamera-motion>

ANEXO 1. Código completo desarrollado para el segundo escenario (detección de movimiento por procesado de imagen).

```
verbose = True
if verbose:
    print "Cargando librerias..."
else:
    print "El detalle de salida a sido deshabilitado"

import picamera
import picamera.array
import datetime
import time
from PIL import Image
from PIL import ImageFont
from PIL import ImageDraw
from fractions import Fraction

#Constantes
SECONDS2MICRO = 1000000  # Constante de conversion de velocidad de obturacion
en segundos a microsegundos

# Parametro configurables por el usuario
imageDir = "images"
imagePath = "/home/pi/Raspy/Motion/Final/" + imageDir
imageNamePrefix = 'capture-'  # Prefijo para todas los nombres de las imagenes.
imageWidth = 1980
imageHeight = 1080
imageVFlip = True   # Volteo vertical de la imagen
imageHFlip = False   # Volteo horizontal de la imagen
imagePreview = False

numberSequence = False

threshold = 10  # Cuanto debe cambiar un pixel. valor entre 1 y 254
sensitivity = 100  # Cuantos pixeles deben cambiar antes de detectar movimiento

nightISO = 800
nightShutSpeed = 6 * SECONDS2MICRO   # Conversion de tiempos de segundo a
constante de microsegundos

# Configuracion avanzada. Preferiblemente no tocar.
testWidth = 128
testHeight = 80
def checkImagePath(imagedir):
    # Encuentra el camino del script python y establece algunas variables
    mypath=os.path.abspath(__file__)
    baseDir=mypath[0:mypath.rfind("/")+1]
    baseFileName=mypath[mypath.rfind("/")+1:mypath.rfind(".")]

    # Configuracion de la ruta de la imagen y crea el directorio si no existe
    imagePath = baseDir + imagedir  # Donde guardar las imagenes
```

```
    # Si la ruta de la imagen no existe. Crea la carpeta
    if not os.path.isdir(imagePath):
        if verbose:
            print "%s - Directorio no encontrada." % (progName)
            print "%s - Creando directorio. %s " % (progName, imagePath)
        os.makedirs(imagePath)
    return imagePath
def takeDayImage(imageWidth, imageHeight, filename):
    if verbose:
        print "takeDayImage - Working ....."
    with picamera.PiCamera() as camera:
        camera.resolution = (imageWidth, imageHeight)
        # camera.rotation = cameraRotate #Nota, usa las variables imageVFlip y
imageHFlip
        if imagePreview:
            camera.start_preview()
        camera.vflip = imageVFlip
        camera.hflip = imageHFlip
        # Modo de dia automatico
        camera.exposure_mode = 'auto'
        camera.awb_mode = 'auto'
        camera.capture(filename)
    if verbose:
        print "takeDayImage - Captured %s" % (filename)
    return filename
def takeNightImage(imageWidth, imageHeight, filename):
    if verbose:
        print "takeNightImage - Working ....."
    with picamera.PiCamera() as camera:
        camera.resolution = (imageWidth, imageHeight)
        if imagePreview:
            camera.start_preview()
        camera.vflip = imageVFlip
        camera.hflip = imageHFlip
        #Ajuste de luz de noche tienen largos tiempos de exposicion
        #Configuracion para condicion de luz baja
        #Establecer una velocidad de 1/6 fps, entonces coloque obturador
        #Velocidad de 6s y ISO aproximadamente a 800 para nightISO
        camera.framerate = Fraction(1, 6)
        camera.shutter_speed = nightShutSpeed
        camera.exposure_mode = 'off'
        camera.iso = nightISO
        #Darle a la camara un largo tiempo para medir el balance de blancos
(AWB)
        time.sleep(10)
        camera.capture(filename)
    if verbose:
        print "checkNightMode - Captured %s" % (filename)
    return filename
def takeMotionImage(width, height, daymode):
    with picamera.PiCamera() as camera:
        time.sleep(1)
        camera.resolution = (width, height)
        with picamera.array.PiRGBArray(camera) as stream:
            if daymode:
                camera.exposure_mode = 'auto'
                camera.awb_mode = 'auto'
            else:
```

```
                #Tomar imagenes con baja luz
                #Establecer una velocidad de fotograma de 1/6 fps, entonces
coloque obturador
                # Velocidad a 6s y ISO a 800
                camera.framerate = Fraction(1, 6)
                camera.shutter_speed = nightShutSpeed
                camera.exposure_mode = 'off'
                camera.iso = nightISO
                #Darle a la camara un largo tiempo para medir el balance de
blancos (AWB)
                time.sleep( 1 )
            camera.capture(stream, format='rgb')
            return stream.array

def scanIfDay(width, height, daymode):
    data1 = takeMotionImage(width, height, daymode)
    while not motionFound:
        data2 = takeMotionImage(width, height, daymode)
        pCnt = 0L;
        diffCount = 0L;
        for w in range(0, width):
            for h in range(0, height):
                #Obtiene la diferencia del pixel
                diff = abs(int(data1[h][w][1]) - int(data2[h][w][1]))
                if  diff > threshold:
                    diffCount += 1
            if diffCount > sensitivity:
                break;
        if diffCount > sensitivity:
            motionFound = True
        else:
             data2 = data1
    return motionFound
def scanMotion(width, height, daymode):
    motionFound = False
    data1 = takeMotionImage(width, height, daymode)
    while not motionFound:
        data2 = takeMotionImage(width, height, daymode)
        diffCount = 0L;
        for w in range(0, width):
            for h in range(0, height):
                #Obtiene la diferencia del pixel
                diff = abs(int(data1[h][w][1]) - int(data2[h][w][1]))
                if  diff > threshold:
                    diffCount += 1
            if diffCount > sensitivity:
                break;
        if diffCount > sensitivity:
            motionFound = True
        else:
            data2 = data1
    return motionFound

def getFileName(imagePath, imageNamePrefix, currentCount):
    rightNow = datetime.datetime.now()
    if numberSequence :
        filename = imagePath + "/" + imageNamePrefix + str(currentCount) +
".jpg"
    else:
```

```
        filename = "%s/%s%04d%02d%02d-%02d%02d%02d.jpg" % ( imagePath,
imageNamePrefix ,rightNow.year, right$
    return filename

def motionDetection():
    print "Escaneando Motion para threshold=%i sensitivity=%i ......" %
(threshold, sensitivity)
    isDay = True
    currentCount= 1000
    while True:
        if scanMotion(testWidth, testHeight, isDay):
            filename = getFileName(imagePath, imageNamePrefix, currentCount)
            if numberSequence:
                currentCount += 1
            if isDay:
                takeDayImage( imageWidth, imageHeight, filename )
            else:
                takeNightImage( imageWidth, imageHeight, filename )

if __name__ == '__main__':
    try:
        motionDetection()
    finally:
        print ""
        print "+++++++++++++++"
        print "Exiting Program"
        print "+++++++++++++++"
        print ""
```